DERIVING OPTIMAL SOLUTIONS FROM INCOMPLETE KNOWLEDGE BASES

THESIS

Shawn Arnold Northrop
Captain, USAF

AFIT/GCS/ENG/95D-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/95D-08

DERIVING OPTIMAL SOLUTIONS FROM INCOMPLETE KNOWLEDGE BASES

THESIS

Shawn Arnold Northrop
Captain, USAF

AFIT/GCS/ENG/95D-08

AFIT/GCS/ENG/95D-08

DERIVING OPTIMAL SOLUTIONS FROM INCOMPLETE KNOWLEDGE BASES

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Science

Shawn Arnold Northrop, B.S.
Captain, USAF

December 1995

Acknowledgements

I would like to dedicate this work to my parents Jack and Sandra Northrop, my sister Debra Nissen and her husband Andrew, whose constant encouragement allowed me to make it through the many tough times (and to this day still don't know what artificial intelligence is).

Many thanks are extended to: Committee members Dr. Eugene Santos, Dr. Richard Deckro, and Dr. Henry Potoczny, and also professors Lt Col James Moore and Maj Price Smith. Howard "Duck" Gleason and Stan Kinderknecht for enduring my endless list of C++ and LaTeX questions, and entertaining me with their discussions on life, the Star Trek universe, and just about everything in between. Ed Williams for his insights and oversights into problem solving and surviving the AFIT experience (he did come back, you know). Linda, Steve, Tracy, Kevin, and the rest of the Food For Thought (FFT) gang for kidnapping me from time to time, and giving me the greatly needed "Calgon bath" treatment. My fellow volleyballers, scattered throughout Kettering and Centerville, who allowed me to vent my frustrations and get a good workout in the process. The other GCS/GCE/GE lackeys that toiled by my side, and whose wise-cracks and antics made the whole experience bearable.

This work was sponsored in part by AFOSR Grant #94-0006.

Shawn Arnold Northrop

Table of Contents

List of Figures

Figure Page

AFIT/GCS/ENG/95D-08

Abstract

Many real world domains can not be represented using Bayesian Networks due to the need for complete probability tables and acyclic knowledge. However, Bayesian Knowledge Bases (BKBs) are a viable method for representing these incomplete domains, but very little research has been performed on inferencing with them. This paper presents three inference engines for extracting optimal solutions from three distinct BKB subclasses: singly-connected, multiply-connected with mutually exclusive cycles, and cyclic. The singly-connected inference engine has a worst case polynomial run time. Performance improvement techniques for increasing inference engine speed are discussed, in addition to a new tool for measuring incompleteness and aiding in BKB Validation & Verification.

DERIVING OPTIMAL SOLUTIONS FROM INCOMPLETE KNOWLEDGE BASES

I. Introduction

Several areas of Artificial Intelligence (AI) attempt to aid the human user by answering questions, constructing explanations, and deriving conclusions. In many domains, approximate answers never seem quite good enough. Whether it is diagnosing blood diseases[4] or identifying minerals and selecting drilling sites[9, 19], the optimal answer is usually of the utmost importance. Expert systems have emerged as one of the leading mechanisms for providing these optimal answers, and thus have become one of the user's most valued tools.

Feigenbaum[10], an early pioneer of expert systems technology, has defined an expert system as " ...an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution." An expert system is comprised of four major subareas: the organization of its knowledge, methods to validate that its knowledge is correct, a mechanism to reason over its knowledge, and an interface to communicate with the user (See Figure 1.1). At the heart of the expert system is the knowledge base itself. Clearly, the structure of the knowledge base will profoundly affect how these four subareas are designed and also how they function.

The first knowledge bases used "if-then" type rules[4], since this structure is favored most by human experts. Statements like "All birds can fly" seem sound, but they are filled with exceptions. Penguins, ostriches, emus, hatchlings, and even deceased birds all violate this rule. As the number of exceptions grow, so does the number of "if-then" rules. Soon, the knowledge base becomes so large and

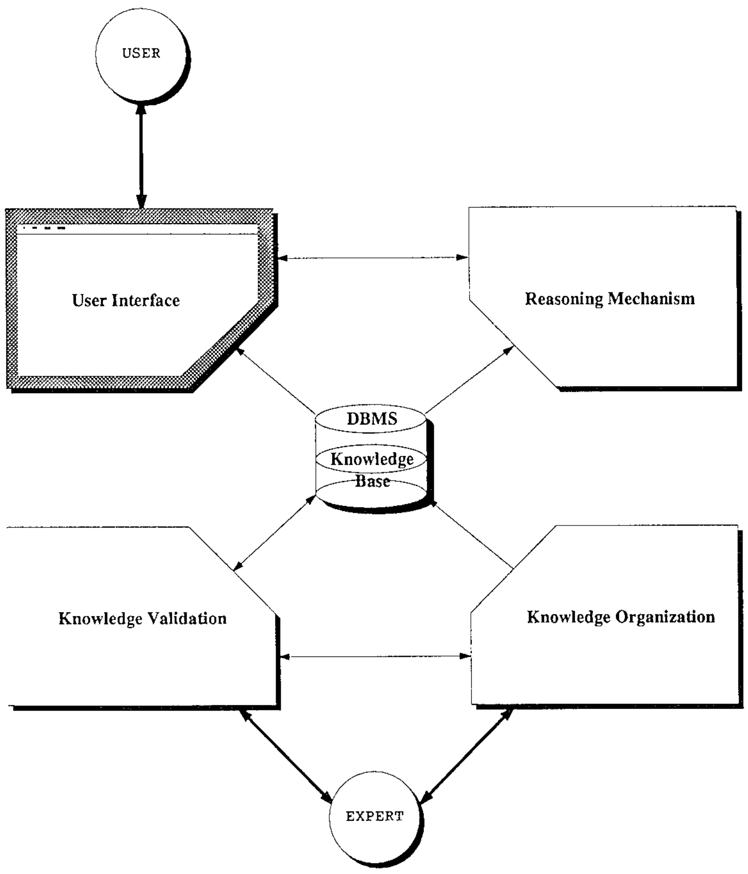


FIG. 1.1. The four major subsystems of an expert system[26].

complex, inferencing becomes nearly impossible. Thus, "if-then" rules do a poor job at handling exceptions and the other uncertain events of the world.

In response to the shortcomings of "if-then" rule models, Bayesian Networks[18, 21, 23] were developed. Their graphical representation and strong roots in probability theory allow them to handle uncertainty with greater ease. However, they have two major shortfalls, the inability to represent any form of cyclic knowledge, and the requirement of complete probability tables before any inferencing can begin. In a small network with just two states and 10 random variables (r.v.s.), $2^{10} = 1024$ probability entries are needed. Assumptions on independence between r.v.s. can markedly reduce the number probabilities, however in large networks, the number of required probabilities can become staggering. Within many real world domains, a number of these probabilities are unknown, thus incomplete knowledge makes maintaining complete probability tables untenable. Similarly, cyclic knowledge permeates many real world domains, a classic example being the rules "If fire, then smoke", and "If smoke, then fire". In order to represent this type of relation, the Bayesian Network must also be cyclic. Current algorithms attempting to inference over a cyclic Bayesian Network are unable to distinguish between acyclic solutions, which are acceptable, and cyclic solutions, which are unacceptable, thus forcing the requirement of an acyclic graph. Despite the fact that Bayesian Networks are relatively straightforward to implement, these two deficiencies, cyclicity and incompleteness, severely limit their ability to represent many real world knowledge domains.

One proposed solution to incomplete probability tables was offered by Ramoni and Riva[24]. Their method, Ignorant Belief Networks (IBNs), uses the same constructs as a Bayesian Network, i.e., a directed acyclic graph and a probability table. However, the probability table begins with probability intervals in the known entries, and blanks in the unknown entries. After a refining process, a complete probability distribution is derived. Ramoni and Riva[24] never clearly state how much incompleteness this approach can regenerate, nor the minimum number of probabilities needed to apply their method. This coupled with its need for complete probability tables, either derived explicitly or implicitly, and its inability to express cyclic knowledge, considerably dampens the appeal of this method.

The other approach that attempts to conquer incompleteness is the Bayesian Knowledge Base (BKB)[2, 27]. BKBs differ greatly from both Bayesian Networks and IBNs in that the latter methods represent knowledge at the random variable (r.v.) level. BKBs break r.v.s. down and represent them at the instantiation level. This results in BKBs being more flexible than both Bayesian Networks and IBNs since they can represent some types of cyclic knowledge and can inference with incomplete probability tables. BKBs are also known as Bayesian Forests or Bayesian Multi-Nets[1][1, 2, 27].

With BKBs appearing to be the logical choice for a knowledge representation, there are several inferencing algorithms previously used for Bayesian Networks that could serve as candidates to perform the reasoning task. Traditional graph search techniques, such as Best-First search and A^*[22], have been proven to find the optimal solution, however they can run in exponential time in the worst case. An apparent improvement is to convert the Bayesian Network to a Weighted AND/OR Directed Acyclic Graph (WAODAG) and find the minimal cost proof using the same Best-First search techniques[5, 15], or by further reducing the WAODAG to a set of 0-1 linear inequalities and then using the Simplex method[14] in conjunction with branch and bound techniques[1, 31]. This latter method appears promising, exhibiting an expected case polynomial time growth rate [1, 29, 31]. The major drawback to using a WAODAG to solve a Bayesian Network is despite the fact that the conversion can be performed in linear time, the number of nodes in the resulting WAODAG is exponential with respect to the size of the original Bayesian Network[18]. Message passing schemes used by [23, 35, 36] are the fastest running algorithms to date, guaranteeing a worst-case polynomial run time, however, only on singly-connected Bayesian Networks[35, 36]. Lastly, an algorithm for solving a Bayesian Network by reducing it to an optimal ordering of factors has been presented, but its complexity is $O(2^n)$ even in singly-connected networks[18].

This work is in support of the integration of probability theory with expert systems into a complete framework called Probabilities, Expert Systems, Knowledge, and Inference (PESKI)[26, 32]. The four major components of the PESKI architecture (Natural Language Interface, Inference Engine, Explanation & Interpretation, and Knowledge Acquisition & Maintenance) are combined into three subsystems: User

[1]Originally dubbed "Bayesian Multi-Nets"[27], the name was changed to distinguish them from the Bayesian multinetworks of Heckerman and Geiger [12].

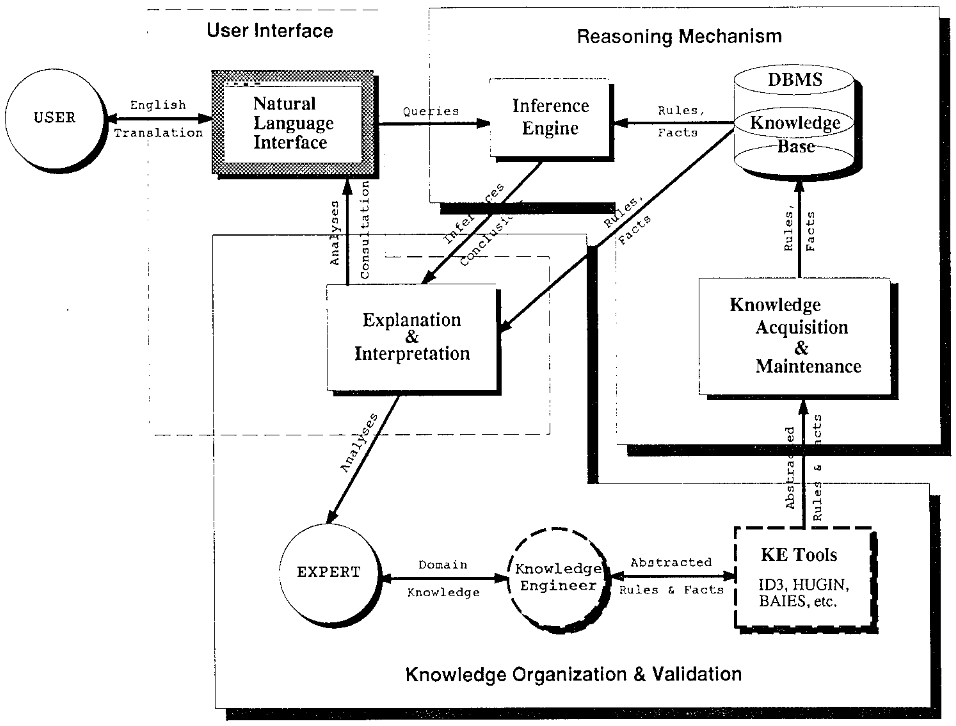


FIG. 1.2. The PESKI architecture[26].

Interface, Knowledge Organization & Validation, and Reasoning Mechanism (See Figure 1.2). The Knowledge Acquisition component of the Organization & Validation subsystem was developed and implemented in [2]. The other components and subsystems are still being researched. When fully integrated, the PESKI architecture can benefit many domains by helping a user make decisions, plan, and solve problems. One such domain is NASA's Post-Test Diagnostic System for the Space Shuttle's Main Engines[2, 3], where preliminary knowledge acquisition tasks have already begun to acquire this domain into the BKB knowledge structure[2].

This paper concentrates on the Reasoning Mechanism subsystem, in particular, the inference engine algorithms that will be used to answer queries from the user, and also aid in the Organization and Validation of the knowledge base. Chapter 2 discusses the background necessary to comprehend BKBs and their relative probabilistic semantics. Chapter 3 presents three inference engines for extracting optimal solutions from various classes of BKBs, plus additional optimization techniques to increase speed. Chapter 4 addresses modifications that can be performed on the inferences engines in Chapter 3 to aid in the Organization and Validation process.

II. Background

Bayesian Knowledge Bases are very similar in some respects to their cousins, Bayesian Networks. However, the greatest difference is in one key area: Bayesian Networks express dependencies between r.v.s., whereas BKBs express dependencies between instantiations of those same r.v.s..

For example, examine Figure 2.1, a simple Bayesian Network. By decomposing the r.v.s. into their respective instantiations and merging in the associated probability table, we can produce Figure 2.2, the corresponding BKB. The nodes labelled A=a0 and A=a1, called instantiation nodes (I-nodes), are the instances of A, and the blackened circles, called support nodes (S-nodes), house the probabilities once held in the probability table of the Bayesian Network. The in-bound arcs of a S-node denote the *tail* dependencies for the probability entry, and the out-bound arc denotes the *head* I-node that receives both the dependencies and the associated probability. For example, S-node x5 represents $\mathrm{P(C = c0|A = a0, B = b0)}$. An S-node can have zero or more in-bound arcs, but must have exactly one out-bound arc. S-nodes are treated as AND nodes, i.e., in order to *clamp* an S-node, i.e. select it for inclusion in a solution, every in-bound I-node must also be clamped. I-nodes are treated as exclusive-OR nodes, i.e., in order to clamp an I-node, at most one in-bound S-node must be clamped. Also, a single I-node must be clamped for every r.v., thus denoting the r.v.'s state.

A careful eye will notice that Figure 2.2 is missing S-nodes for the probabilities

$$\mathrm{P(C = c0|A = a0, B = b1)}$$
$$\mathrm{P(C = c0|A = a1, B = b0)}$$
$$\mathrm{P(C = c1|A = a0, B = b1)}$$
$$\mathrm{P(C = c1|A = a1, B = b0)}$$

Thus, the BKB is said to be *incomplete*, since it only has eight of its full complement of twelve probabilities. BKBs used to date[2] appear to be missing probabilities for

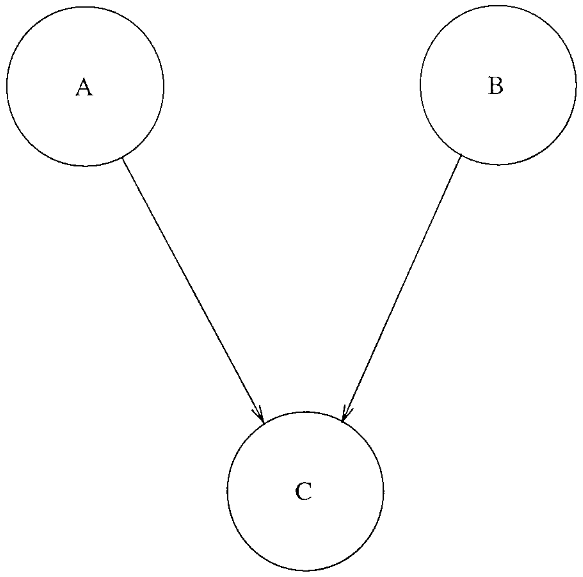


FIG. 2.1. A simple Bayesian Network

nearly every r.v.. However, some r.v.s. may have a complete probability set, which demonstrates a better understanding by the knowledge engineer for those particular r.v.s.. Above all, the BKB must contain the necessary probabilities to answer the majority, if not all, of the posed queries even though the BKB is incomplete. The question of how many probabilities are necessary is an issue for Knowledge Validation and Verification, and is beyond the scope of this research.

Unlike Bayesian Networks, we can still inference over the BKB despite incompleteness. Inferencing is performed by computing the joint probability distribution (JPD) by clamping one I-node for each r.v. in the BKB, and calculating the associated probability. This is still accomplished by using Bayes Theorem [23], in this case

$$\mathrm{P(A, B, C) = P(C|A, B)P(A)P(B)}$$

Within any BKB, there are a combinatorial number of JPDs, the maximum JPD being the resultant clamping that yields the highest probability. Computing the maximum JPD is NP-hard, as stated in Theorem 2.0.1.

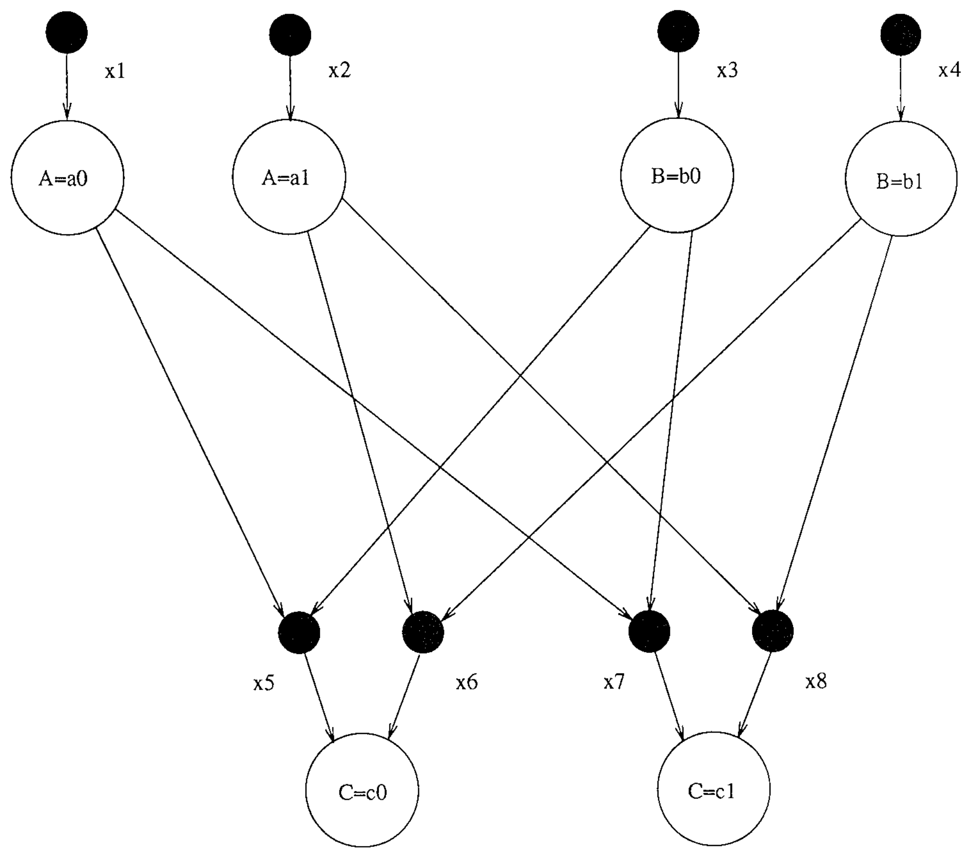


FIG. 2.2. A simple Bayesian Knowledge Base

THEOREM 2.0.1. *Computing the maximum joint probability distribution in a Bayesian Knowledge Base is NP-hard.*

Proof. It is obvious that any Bayesian Network can be converted to a BKB. Since a multiply-connected Bayesian Network has been proven to be NP-hard[6, 33], it follows that a BKB is also NP-hard[11]. □

In order to calculate a JPD, a BKB must be *consistent.* A BKB is *inconsistent* when there are multiple ways to compute the same JPD with no guarantee that the probabilities are equal [27]. Banks[2] developed eight constraints that guarantee a BKB to be consistent. Constraints 4, 5, and 8 are the most relevant for our research, and are discussed below. All eight constraints are described in-depth in [2].

Constraint 4: Any S-nodes which share the same head I-node must be mutually exclusive [27]. Two S-nodes are *mutually exclusive* when there exists two different I-nodes in the tail of both S-nodes, each of which belongs to the same r.v..

Constraint 5: Given a particular inference chain, an S-node's head cannot re-occur in the tails of its successors in that chain, unless prevented by a mutual exclusive condition[2]. In short, no logical cycles are allowed unless a mutual exclusive condition exists in the cycle. The following equations form such an inference chain:

$$\begin{aligned}
\mathrm{P(A = a0|B = b5, M = m1)} &= .45 \\
\mathrm{P(B = b5|C = c1)} &= .30 \\
\mathrm{P(C = c1|D = d4, M = m0)} &= .90 \\
\mathrm{P(D = d4|A = a0)} &= .50
\end{aligned}$$

The two different I-nodes belonging to r.v. M provide the necessary mutual exclusive condition, without which, this logical cycle would not be allowed.

Constraint 8: S-nodes belonging to same r.v. that are not mutually exclusive (MUTEX) cannot have the sum of their probabilities exceed 1.

During the course of our research, we discovered that Constraint 5 is not a necessary condition. An inference engine in Chapter 3 was developed that can reason over BKBs with cycles that violate Constraint 5. Not only was this inference engine necessary to utilize our AS-nodes in Chapter 4 , but it also removes the burden on Knowledge Acquisition to identify and eliminate the non-MUTEX cycles prohibited by Constraint 5. References to Constraint 5 no longer being a necessary condition for BKB consistency will be found throughout this paper.

III. BKB Inference Engines

BKBs have a very unique structure. Each I-node is surrounded by S-nodes, and vice versa for S-nodes, i.e., I-nodes are never adjacent to I-nodes, and S-nodes are never adjacent to S-nodes. We exploited this structure during the construction of our inference engines in order to increase their speed. To this end, a BKB can fall into one of three distinct subclasses, each of which has its own specialized inference engine. The first inference engine can only be used on a certain type of singly-connected BKB, and has a worst case polynomial run time. The other two inference engines can both be used on multiply-connected BKBs, the latter handling cyclic BKBs. Both of these exhibit expected case polynomial time growth rates, however, if no solution exists, they will run in exponential time while exhausting the entire solution space.

3.1 BKB Polytrees

This inference engine can only be applied to our first BKB subclass, BKB Polytrees.

Definition 3.1.1. *A BKB Polytree is a BKB which any two r.v.s., there exists a single r.v. path between them.*

Definition 3.1.2. *A r.v. path is a connected path between two nodes such that the path is only composed of r.v.s.*[1].

An example of a BKB Polytree can be seen in Figure 3.1. Even though it appears disconnected at the instantiation level, the dependencies still hold at the r.v. level. This is evident by examining r.v. D, which is dependent on both r.v.s. X and C.

[1]In this definition, a r.v. is the union of its I-nodes and S-nodes.

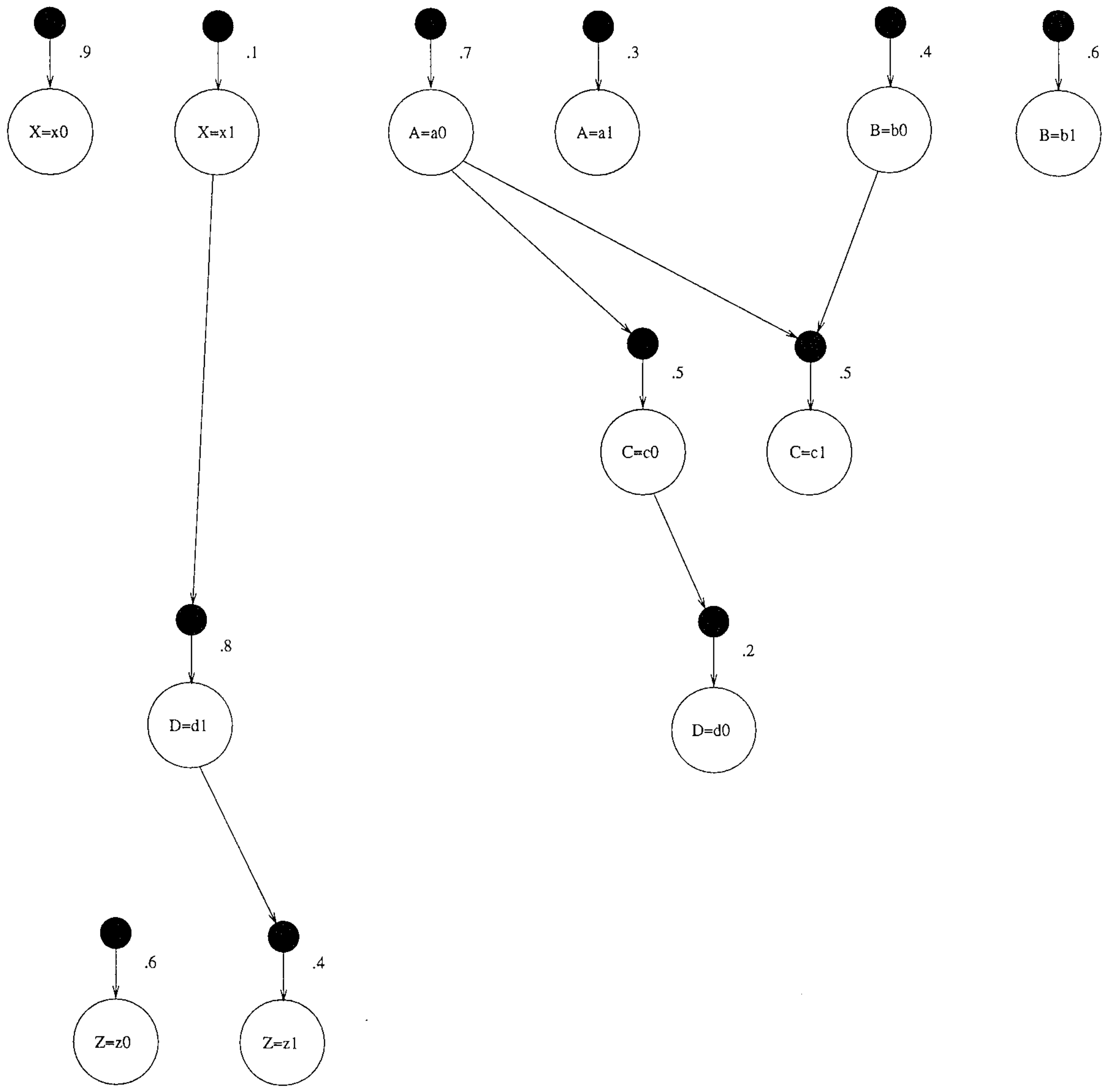


FIG. 3.1. A BKB Polytree

The main strategy of this algorithm[2] is to designate a r.v.[3] in the BKB as a sink and to propagate the minimal amount of necessary probabilities to the sink via a set of unique paths to cover all the r.v.s. in the BKB.

In Figure 3.1, if we were to select r.v. A as the sink, there exists three unique paths to A in the BKB, namely $B \rightarrow C \rightarrow A$, $Z \rightarrow D \rightarrow C \rightarrow A$, and $X \rightarrow D \rightarrow C \rightarrow A$. Nötice that every path originates at a root or leaf node, and terminates at the sink. By passing the appropriate probabilities along these paths, we can compute the maximum JPD, or Max $P(A, B, C, D, X, Z)$.

The path to the sink can be constructed by performing a Breadth-First *dependency* search, and caching the results. Using the term *dependency* is a slight misnomer, since we really mean all of the r.v.s. a selected r.v. depends on, and all of the r.v.s. that are dependent on the selected r.v..

The algorithm requires a queue, Q, to temporarily hold dependent r.v.s., a set, V, to hold the r.v.s. already visited, and a stack, P, to hold the paths back to the sink. The path construction algorithm is as follows:

```
Enqueue the sink to Q
WHILE (Q $\neq \emptyset$)
    Dequeue $s$ from Q
    V = V $\cup$ {$s$}
    D = {the set of parent and child r.v.s of $s$} − V
    Push a separate arc from $s$ to each member of D onto P
    Enqueue each member of D to Q
END WHILE
```

The path construction algorithm takes the same amount of time to run as a typical Breadth-First search[7], $O(|V| + |E|)$, where V is the set of all I-nodes and S-nodes, and E is the set of all edges in the BKB. The path, P, gives a road map to allow

[2]This algorithm adopts a message passing scheme similar to [35], thus many of the definitions, theorems, and proofs will bear a close resemblance to those found is Sy's work.

[3]The set of its I-nodes.

probability streams, also called message streams, to be passed from the outer leafs and roots of the BKB and arrive at the sink.

Passing the probabilities of a r.v. b to a r.v. a is designated by $\mathrm{M}_{\mathrm{b}\rightarrow\mathrm{a}}$. If b is a child of a, then we need only send the maximum probabilities that are dependent on the receiving r.v., a, designated $\mathrm{Max}_{\mathrm{a}}[\mathrm{P}(\mathrm{b}|\mathrm{a})]$. For example in Figure 3.1, $\mathrm{M}_{\mathrm{Z}\rightarrow\mathrm{D}} = [\mathrm{P}(\mathrm{Z} = \mathrm{z0}) = .6]$. Since $[\mathrm{P}(\mathrm{Z} = \mathrm{z0}) = .6]$ is shorthand for $[\mathrm{P}(\mathrm{Z} = \mathrm{z0}|\mathrm{D} = \mathrm{d0}) = .6, \mathrm{P}(\mathrm{Z} = \mathrm{z0}|\mathrm{D} = \mathrm{d1}) = .6]$, and $\mathrm{P}(\mathrm{Z} = \mathrm{z0}|\mathrm{D} = \mathrm{d1}) = .6 \geq \mathrm{P}(\mathrm{Z} = \mathrm{z1}|\mathrm{D} = \mathrm{d1}) = .4$, only $\mathrm{P}(\mathrm{Z} = \mathrm{z0}) = .6$ is passed. If b is a parent of a, then we need only send the maximum probabilities of the passed r.v., b, designated $\mathrm{Max}_{\mathrm{b}}[\mathrm{P}(\mathrm{b})]$. An example is $\mathrm{M}_{\mathrm{X}\rightarrow\mathrm{D}} = [\mathrm{P}(\mathrm{X} = \mathrm{x0}) = .9, \mathrm{P}(\mathrm{X} = \mathrm{x1}) = .1]$. These are the minimal number of probabilities needed to calculate the maximum JPD. When these messages arrive at r.v. D, we must combine them with the probabilities of r.v. D using an operation called a convolution, denoted $*$. When all of the probabilities arrive at a r.v., the result is called a belief vector.

DEFINITION 3.1.3. *Given* $M_{d\rightarrow x} = [m_{d\rightarrow X_1}, m_{d\rightarrow X_2}, \cdots, m_{d\rightarrow X_n}]$, *and* $P(x|J_x) = [P(X_1|v1_x), P(X_1|v2_x), \cdots, P(X_n|vk_x)]$ *(where* vi_x *are the I-nodes of the r.v.s. in* J_x*), the convolution of* $M_{d\rightarrow x}$ *with* $P(x|J_x)$ *is defined as the product of every single term in* $P(x|J_x)$ *with a consistent* $m_{d\rightarrow x}$ *in* $M_{d\rightarrow x}$*; where* J_x *is the set of immediate parent r.v.s. of* x *in a BKB.* $P(x|J_x)$ *and* $m_{d\rightarrow x}$ *are consistent with each other if the I-node of* x *in* $m_{d\rightarrow x}$ *and that in* $P(x|J_x)$ *are identical.*

DEFINITION 3.1.4. *A Belief Vector of a node x, Bel(x), is defined as the convolution of all* $M_{d\rightarrow x}$ *with* $P(x|J_x)$ *such that* $Bel(x) = M_{d_1\rightarrow x} * M_{d_2\rightarrow x} * \cdots * M_{d_k\rightarrow x} * P(x|J_x)$ *where* d_i *are the r.v.s. which propagate* $M_{d_i\rightarrow x}$ *to* x *for* $i = 1 \ldots k$.

Using our definitions of convolution and belief vector, we can now calculate Bel(D).

$$\begin{aligned}
\mathrm{Bel(D)} &= \mathrm{M_{X\rightarrow D} * M_{Z\rightarrow D} * P(D|X,C)} \\
&= \mathrm{M_{X\rightarrow D}} * \left[\begin{array}{lcl} \mathrm{P(Z=z0)} & = & .6 \end{array}\right] * \left[\begin{array}{lcl} \mathrm{P(D=d0|C=c0)} & = & .2 \\ \mathrm{P(D=d1|X=x1)} & = & .8 \end{array}\right] \\
&= \mathrm{M_{X\rightarrow D}} * \left[\begin{array}{lcl} \mathrm{P(D=d0,Z=z0|C=c0)} & = & .12 \\ \mathrm{P(D=d1,Z=z0|X=x1)} & = & .48 \end{array}\right] \\
&= \left[\begin{array}{lcl} \mathrm{P(X=x0)} & = & .9 \\ \mathrm{P(X=x1)} & = & .1 \end{array}\right] * \left[\begin{array}{lcl} \mathrm{P(D=d0,Z=z0|C=c0)} & = & .12 \\ \mathrm{P(D=d1,Z=z0|X=x1)} & = & .48 \end{array}\right] \\
&= \left[\begin{array}{lcl} \mathrm{P(D=d0,X=x0,Z=z0|C=c0)} & = & .108 \\ \mathrm{P(D=d0,X=x1,Z=z0|C=c0)} & = & .012 \\ \mathrm{P(D=d1,X=x1,Z=z0)} & = & .048 \end{array}\right]
\end{aligned}$$

$\mathrm{M_{D\rightarrow C}}$ can now be formulated as $\mathrm{Max_C[Bel(D|C)]}$, where

$$\mathrm{Max_C[Bel(D|C)]} = \left[\begin{array}{lcl} \mathrm{P(D=d0,X=x0,Z=z0|C=c0)} & = & .108 \\ \mathrm{P(D=d1,X=x1,Z=z0)} & = & .048 \end{array}\right]$$

With Definitions 3.1.3 and 3.1.4, we can now formalize the message stream $\mathrm{M_{b\rightarrow a}}$ in the following definition:

DEFINITION 3.1.5. *A message stream that a r.v. b propagates to a r.v. a in a BKB is defined as*

$$M_{b\rightarrow a} = \begin{cases} Max_a[Bel(b|a)] \\ \quad if\ a\ is\ an\ immediate\ parent\ of\ b \\ Max_b[Bel(b)] \\ \quad if\ b\ is\ an\ immediate\ parent\ of\ a \end{cases}$$

where

$$Bel(b|a) = \begin{cases} M_{d_1\rightarrow b} * \cdots * M_{d_k\rightarrow b} * P(b|a, p_1 \ldots p_k) \\ \quad if\ a\ is\ an\ immediate\ parent\ of\ b \end{cases}$$

$$Bel(b) = \begin{cases} M_{p_1\rightarrow b} * \cdots * M_{p_k\rightarrow b} * P(b|p_1 \ldots p_k) \\ \quad if\ b\ is\ an\ immediate\ parent\ of\ a \end{cases}$$

$d_1 \ldots d_k$ are the immediate child r.v.s. of b, and $p_1 \ldots p_k$ are the immediate parent r.v.s. of b

Using our path construction algorithm and Definition 3.1.5, we can now state a formal algorithm for finding an optimal solution to a BKB Polytree:

```
Accept evidence, E
Choose r.v. S as sink and build path P for the propagation of message streams

DO
    Pop P and compose $M_{b \to a}$ using Definition 3.1.5
    Instantiate any probabilities in $M_{b \to a}$ using E

    Perform a convolution operation to combine the incoming proba-
    bilities within $M_{b \to a}$ and update Bel(a) at each r.v. traversal

    IF a convolution operator yields an empty belief vector
        Exit failing with no solution
    END IF
UNTIL (P $\neq \emptyset$)

The optimal solution is $\mathrm{Max}[\mathrm{Bel(S)}]$.
```

If all of the r.v.s. in the BKB are not visited after performing this algorithm, simply re-apply it on the remaining r.v.s. and combine the results. The heart of the algorithm relies on the sufficiency and completeness of the probabilities contained in $M_{b \to a}$, and is summarized in Theorem 3.1.1. The algorithm has the same worst case complexity as [35] which is, $O(kn)$, where k is the length of the longest path in the BKB, and n is the maximum number of r.v. states — defined as the product of the number of S-nodes of a r.v. and the number of parent and child r.v.s. of the r.v.[35].

THEOREM 3.1.1. $M_{b \to a}$ *carries sufficient and complete information for the computation of the largest* $P(p1|J_{p1}) \cdots P(pn|J_{pn})P(b|J_b)P(d1|bJ_{d1}) \cdots P(dm|bJ_{dm})$*; where p1, p2, …, pn are the parent nodes of b, and d1, d2, …, dm are the child nodes of b.*

Proof. Without the loss of generality, suppose p1, p2, …, pn and d1, d2, …, dm form two paths to propagate message streams to b, i.e., $p1 \to p2 \to \cdots \to pn \to b$ and $d1 \to d2 \to \cdots \to dm \to b$.

$M_{p1 \to p2}$ carries $P(p1)$ for p1 = all possible values of p1, and $M_{p2 \to p3}$ carries $\mathrm{Max}_{p2}[P(p1)P(p2|J_{p2})]$ for every possible value of p2, and so forth according to Definition 3.1.5. When the message stream carrying all pi reaches b, we have $\mathrm{Max}[P(p1)P(p2|J_{p2}) \cdots P(pn|J_{pn})P(b|J_b)]$. Similarly, $M_{d1 \rightarrow d2}$ carries $\mathrm{Max}_{d1}[P(d1|J_{d1})]$ for every possible value of d2, $M_{d2 \rightarrow d3}$ carries $\mathrm{Max}_{d2}[P(d1|J_{d1})P(d2|J_{d2})]$ for every possible value of d3, and so forth according to Definition 3.1.5. When the message streams carrying all di reach b and are combined with all pi, we get, in a general form, $\mathrm{Max}[P(p1|J_{p1}) \cdots P(pn|J_p n)P(b|J_b)P(d1|J_{d1}) \cdots P(dm|J_{dm})]$ for every possible value of b. From here we can see that one of the $\mathrm{Max}_b[\bullet]$ must be the largest of $\mathrm{Max}[P(pn|J_{pn}) \cdots P(d1|J_{p1}) \cdots P(b|J_b)) \cdots P(dm|J_{dm})]$, or else $\mathrm{Max}_b[\bullet] = \emptyset$, in which case there is no solution. $\Box$

3.2 General BKBs

This inference engine can be applied to multiply-connected BKBs, including BKB Polytrees, but does not work on BKBs containing cycles that are not MUTEX (See Constraint 5). The algorithm is based on those employed in [29, 31].

Our algorithm begins by converting a BKB to a series of 0-1 linear inequality constraints and then uses the Simplex method[14] in conjunction with branch and bound techniques[14] to find an optimal solution[29, 31]. This algorithm, based on the Land-Doig[17] approach to integer linear programming (ILP), was chosen because of its fast run time, the requirement that the graph be of the AND/OR variety[4], and the ability to specialize the algorithm for even greater performance.

In order to convert a BKB into a series of linear inequality constraints, every node[5] in the BKB is represented by a 0-1 variable. Let x_n denote a variable representing the node n. Let every variable contain the value 1 if the node is clamped and 0 if it is not clamped. For example, if n is clamped, $x_n = 1$ and if n is not clamped $x_n = 0$. Also, let D_n represent the set of all nodes that are in the tail of node n, and I_n represent the set of all instantiations of a node n.

[4]A BKB is an AND/Exclusive-OR graph.

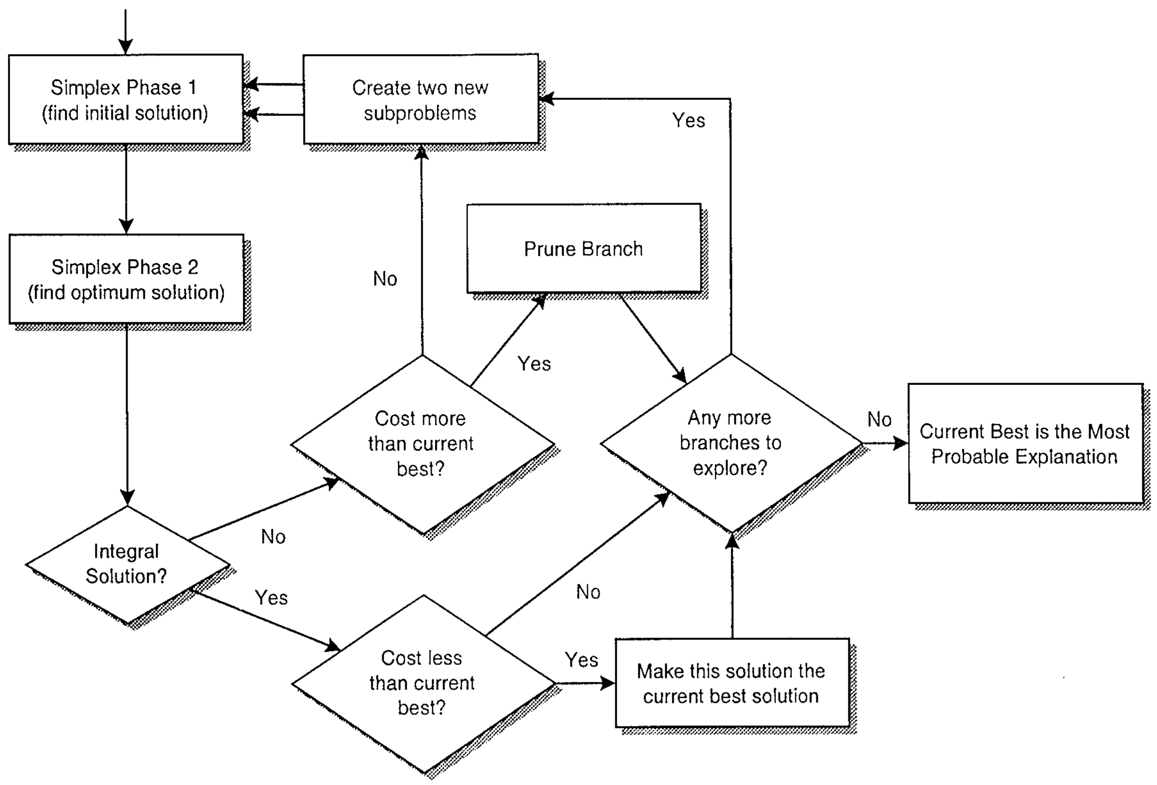
[5]A *node* can be either an I-node or S-node.

After modifying and simplifying the constraints in [1, 31], the constraints for a BKB are as follows:

1. If q is an S-node,
$$|D_q|x_q - \sum_{p \in D_q} x_p \leq 0 \tag{3.1}$$
2. If q is an I-node,
$$x_q - \sum_{p \in D_q} x_p \leq 0 \tag{3.2}$$
3. If q is a r.v.,
$$\sum_{p \in I_q} x_p = 1 \tag{3.3}$$
4. If q is an I-node submitted as evidence,
$$x_q = 1 \tag{3.4}$$

A cost function must also be provided. If S_n is the set of all S-nodes in the BKB, and $prob_q$ is the probability associated with an S-node q, the function is represented by
$$\sum_{q \in S_n} \log(prob_q)\ x_q \tag{3.5}$$
A least cost solution is regarded as an optimal solution.

Equation 3.1 represents that an S-node can equal 1 if all of its tail I-nodes equal 1. Equation 3.2 represents that an I-node can only equal 1 if and only if at most one of its S-nodes equals 1. Equation 3.3 represents that one I-node of every r.v. must equal 1. Initially, the Simplex method finds an optimal assignment to the variables that both satisfies the constraints and minimizes the cost function. A two stage process is used. The first phase finds a solution in the solution space that may not necessarily be optimal. The second phase then zeros in and finds the optimal solution. The optimal linear program solution may or may not be *integral*, i.e., a 0-1 solution. If it is integral, the solution is both feasible and optimal to the integer



Simplex Phase 1 (find initial solution)
Create two new subproblems
Yes
Simplex Phase 2 (find optimum solution)
No
Prune Branch
Yes
Cost more than current best?
Any more branches to explore?
No
Current Best is the Most Probable Explanation
No
Integral Solution?
Yes
No
Cost less than current best?
Yes
Make this solution the current best solution

FIG. 3.2. The General BKB Inference Engine[1]

problem, and the inference engine stops. If it is not integral, the inference engine will begin a Branch and Bound procedure, where it branches on non-integral values forcing each variable to 1 and then 0, such that all possible combinations of integral solutions are considered either by fathoming or bounding. Figure 3.3 is a Branch and Bound tree, a graphical representation of the Branch and Bound process. Each branch node is eliminated either through bounding, or solving a Simplex problem. Figure 3.2 provides a detailed account of the entire inferencing process. Research has shown that the entire algorithm exhibits an expected case polynomial time growth rate when solving generalized AND-OR graphs[31].

3.3 Cyclic BKBs

This last inference engine can reason over *ANY* class of BKB, regardless of whether it contains non-MUTEX cycles or is multiply-connected. By including addi-

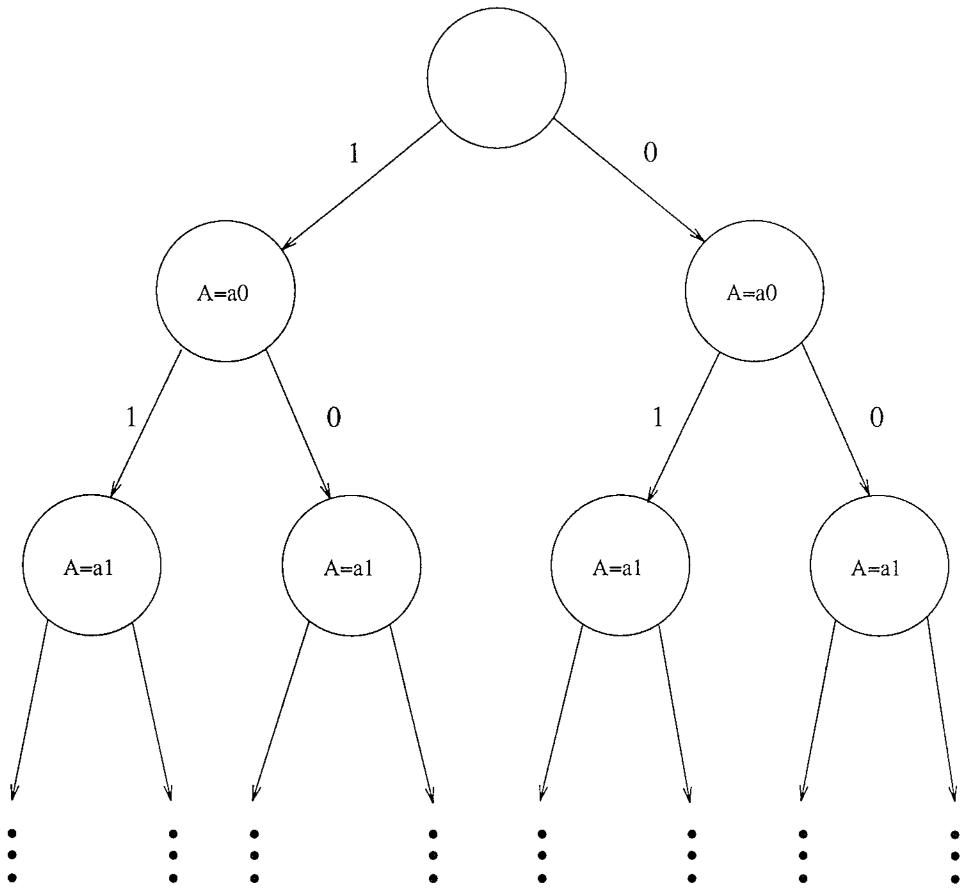


FIG. 3.3. The Branch and Bound tree

tional constraints to the Section 3.2 inference engine, we can eliminate the selection of two paths that would cause a cycle in a solution subgraph.

For a solution subgraph to be acyclic, it must provide a topological ordering of its nodes[30]. If a node p is a dependent tail node of q, then the topological value of p must be less than the value of q. Therefore, we assign a real variable, t_n, to every node n in the BKB to hold n's topological value. We will continue using the same notation as used in Section 3.2, however, let m_{pq} denote the edge between node p and q, and let V denote the number of S-nodes and I-nodes in the BKB. According to [30], the following additional constraints must be added to those of Section 3.2 to inference over a typical AND/OR graph:

1. If q is an S-node,

$$2|\mathrm{V}|(1-\mathrm{x_q})+\mathrm{t_q}\geq \mathrm{t_p}+1,\ \text{for each p}\ \in\ \mathrm{D_q} \tag{3.6}$$

2. If q is an I-node,

$$m_{\mathrm{pq}}\leq \mathrm{x_p},\ \text{for each p}\ \in\ \mathrm{D_q} \tag{3.7}$$

$$2|\mathrm{V}|(1-\mathrm{m_{pq}})+\mathrm{t_q}\geq \mathrm{t_p}+1,\ \text{for each p}\ \in\ \mathrm{D_q} \tag{3.8}$$

The BKB's special structure allows us to further simplify these constraints. Since an S-node has a single I-node in its head, $\mathrm{x_p}$ can be used to identify $\mathrm{m_{pq}}$. Therefore, Constraint 3.7 will always be true, and can be eliminated. Furthermore, if an S-node does not have any I-nodes in its tail, there is no need to create a cyclicity constraint between that S-node and its head I-node. This follows because a cycle only occurs between I-nodes with S-nodes functioning as intermediaries, thus a cycle must pass through an S-node. If an S-node has no tail I-nodes, there is no possibility of it participating in a cycle. Lastly, because an I-node can only be selected if one of its S-nodes is selected and all of the tail I-nodes of that S-node are selected, we can merge Constraints 3.6 and 3.8 into a single constraint.

1. If p is an S-node and $\mathrm{D_p}\neq\emptyset$,

$$2|\mathrm{S}|^2(1-\mathrm{x_p})+\mathrm{t_q}\geq 1+\sum_{\mathrm{r}\in \mathrm{D_p}}\mathrm{x_r} \tag{3.9}$$

where S is the number of S-nodes in the BKB. This constraint simply states that if an S-node is selected, then the topological value of its head I-node must be greater than the sum of the topological values of its tail I-nodes. This constraint enforces the topological ordering necessary to prevent cyclic solutions, eliminates the need for topological variables for the S-nodes, and greatly reduces the total number constraints originally proposed by [30].

Research has shown that this method also exhibits an expected case polynomial time growth rate in generalized cyclic AND-OR graphs[30], however, it is somewhat slower than the inference engine for generalized AND-OR graphs due to the additional constraints. Furthermore, it has the benefit of handling cycles without the need for identifying them explicitly, thus, it is very effective on BKBs with a large number of cycles. However, the number of additional constraints is a multiple of the number of original constraints. For very large BKBs with few non-MUTEX cycles, the number of additional constraints could be excessive. This could adversely affect the relative size of cyclic BKBs this method could inference over.

Another method to prevent cyclic solutions is to explicitly identify every cycle in the BKB, and construct the minimal number of constraints to eliminate these cycles from the solution subgraph. Examples of this method can be found in [30]. Cycle identification can be performed efficiently in $O((|V| + |E|)(n + 1))$ using a Depth-First search algorithm found in [25].

Regardless of which method is used, this inference engine renders Constraint 5 obsolete. The sole purpose of Constraint 5 was to prevent the occurrence of a cycle in the solution subgraph, thus making the solution inconsistent. Since this inferencing method guarantees that all solution subgraphs are acyclic, it also guarantees consistency.

3.4 Performance Improvement Techniques

All three inference engines can benefit greatly by preprocessing the BKB and removing unneeded nodes. By using evidence submitted by the user, we can locally propagate the truth values by attempting to satisfy, and as a result, remove BKB constraints generated by Equations 3.1, 3.2, and 3.3.

Let H_n represent the set of all nodes that are in the head of node n. By using our previous notation, we can use the following pruning rules to reduce a BKB's size:

1. If q is an I-node and $x_q = 1$, then

$$\sum_{p \in I_q - \{q\}} x_p = 0 \tag{3.10}$$

$$\text{Remove Constraint 3.3 containing } x_q \tag{3.11}$$

2. If q is an I-node and $x_q = 0$, then

$$\sum_{p \in D_q} x_p = 0 \tag{3.12}$$

$$\sum_{p \in H_q} x_p = 0 \tag{3.13}$$

$$\text{Remove all Constraints 3.1 and 3.2 containing } x_p \text{ and } x_q \tag{3.14}$$

$$\text{Remove Constraint 3.9 containing } x_p \text{ and } t_q \tag{3.15}$$

3. If q is an I-node and $x_q = 0$ and $|I_q - \{q\}| = 1$, then

$$\sum_{p \in I_q - \{q\}} x_p = 1 \tag{3.16}$$

4. If q is an I-node and $D_q = \emptyset$, then

$$x_q = 0 \tag{3.17}$$

5. If q is a r.v. and $\sum_{p \in I_q} x_p = 0$, then

$$\text{There is no feasible solution.} \tag{3.18}$$

Applying these rules to a BKB will force certain variables to equal 0, thus their corresponding nodes can be pruned from the graph. The rules are most useful on the two Multiply-Connected Inference Engines, where they can preprocess the BKB before the inference engine is evoked, and also at each branch in the Branch and

Input	Receiving Rule
Evidence	Rule 1
Rule 1	Rule 2 Rule 3 Rule 4
Rule 2	Rule 3 Rule 5
Rule 3	Rule 1
Rule 4	Rule 2 Rule 3 Rule 5

TABLE 3.1. Pruning rule dependencies.

Bound tree when additional nodes are clamped[14]. Rule 5 is particularly noteworthy. Since a BKB is incomplete, it is possible to submit evidence that will result in the BKB being unable to return a solution. By preprocessing BKBs submitted to all three inference engines, not only will the computational efficiency increase[8, 16], but these Rules, via Rule 5, may avoid the need to run the inference engine at all. Furthermore, during the Branch and Bound process, if the pruning rules identify a BKB with an infeasible solution, the inference engine can prune off that part of the Branch and Bound tree and immediately backtrack. The reason is that the additional clamped node from the tree caused the infeasible solution, and any further solutions down that branch will include the clamped node, and as a result, will also be infeasible. Since a natural flow exists in the application of these rules, i.e., the results of one rule can be used as the input to another, these rules can be performed in an efficient, step-wise manner. The input/output rule dependencies are shown in Table 3.1.

Another method to increase performance is by using a jumpstart, i.e., using a preliminary solution to help find the optimal solution. In the ILP-based Inference Engines, 60% of the inference engines' time is spent finding an initial solution during phase 1 of the Simplex method, which is not guaranteed to be optimal[28, 31]. By reducing the time required to find this initial solution, the overall time for the algorithm to complete is also reduced. Baenen[1] uses various Depth-First search and local propagation techniques to provide a jumpstart that is close to the optimal solution. Genetic algorithms[20] appear to be another viable jumpstart candidate. Their polynomial run times and ability to close rapidly on the optimal solution seem to make them ideal. However, the number of feasible solutions for any query is not necessarily combinatoric. In fact, there may be no feasible solution for a given query, thus making the task of finding an initial solution very difficult indeed.

The Multiply-Connected Inference Engines can be further improved by using the Polytree Inference Engines during the Branch and Bound procedure. Some BKBs can be forced to take on the structure of a Polytree BKB by identifying certain nodes to be clamped. By clamping these nodes during the Branch and Bound process, we can allow the inference engine to run a Polytree algorithm instead of the Simplex method. The requisite nodes can be identified during knowledge acquisition and stored for later use. A Depth-First search algorithm similar to the one for finding cycles[25] should be sufficient for the task. Since the Polytree Inference Engine has the advantage of *always* producing integral solutions, it should be able to provide an integral solution much quicker than the Simplex method, which produces both integral and non-integral solutions.

Efficiency during the Branch and Bound process can also be dramatically increased by only using the BKB's I-nodes when constructing the Branch and Bound tree. Normally, an ILP will include every variable from the series of linear equality constraints. This includes variables from both S-nodes and I-nodes. Since the optimal solution will contain one I-node from each r.v., it is sufficient to have the solution space include all possible combinations of I-nodes. This is summarized in Theorem 3.4.1. Using only the I-node variable will result in tremendous savings, especially when no feasible solution exists. Since the number of I-nodes is always less than or equal to the number of S-nodes, the path length from the root to a leaf in the Branch and Bound tree will always be cut in half, at a minimum. In addition, by clamping only I-nodes in the tree, we will always get the maximum benefit from our pruning rules, since they are triggered primarily by the clamping of I-nodes.

THEOREM 3.4.1. *Using I-node variables in conjunction with Pruning Rule 3.10 during the Branch and Bound procedure is sufficient to cover all feasible solutions.*

Proof. By only using I-nodes to construct the Branch and Bound tree, each path will constitute a possible I-node solution to any query, and the union of these paths constituting all possible I-node solutions. If the Branch and Bound procedure reaches a leaf, a single I-node for each r.v. will be clamped. Consistency Constraint 4 states that any S-nodes which share the same head I-node must be mutually exclusive. Therefore, applying Pruning Rule 3.10 will leave a single S-node supporting each I-node if a feasible solution exists, or no S-node supporting at least one I-node if no feasible solution exists. If a feasible solution exists, Constraint 3.2 requires each I-node select an S-node, and since each I-node has only one S-node supporting it, the remaining S-nodes are selected. Therefore, all feasible solutions are covered. □

IV. *AS-nodes as an Organization & Validation Inferencing Tool*

During Organization & Validation, it is necessary to evoke the inference engine in order to validate the knowledge base. The incompleteness of a BKB has led to the problem of evaluating the quality of solutions received from the inference engine. When the inference engine returns .000000000037287439, what exactly does that mean and what can we compare it to? What does it mean especially in the presence of incompleteness? Approximating Support Nodes (AS-nodes) are a tool the inference engine can use to find an upper bound on the solution to a query, and thus provide the knowledge engineer a measure of the quality, as well as the incompleteness of that solution. The term upper bound is defined in Definition 4.0.1, however, we must first describe the possible probability distributions (PDs) that will contain an upper bound. These PDs are discussed in Theorem 4.0.2.

THEOREM 4.0.2. *For any incomplete BKB, there exists a set containing an uncountably infinite number of consistent probability distributions (PDs) from which the BKB could be derived.*

Proof. If a BKB is incomplete, it is missing at least one S-node. S-node probabilities must assume a value in the range of [0,1]. Since the subset of real numbers between [0,1] is uncountably infinite, it follows that any continuous subset of real numbers between [0,1] is also uncountably infinite[34]. Therefore, there are an uncountably infinite number of different S-nodes that can be added to the BKB, thus creating an uncountably infinite number of consistent PDs. □

DEFINITION 4.0.1. *Given an incomplete BKB, according to Theorem 4.0.2 there exists a set S containing an uncountably infinite number of consistent PDs from which the BKB could be derived. An identical query i submitted to $\forall s \in S$ will yield a Max JPD for each s. Of these Max JPDs, a single probability p_i will dominate all others. A BKB with AS-nodes is said to deliver a Tight Upper Bound, if for every query* *i, its Max JPD is equal to p_i. A BKB with AS-nodes is said to deliver an Upper Bound, if for every query j, its Max JPD is $p_j \geq p_i$.*

AS-nodes possess an identical structure to S-nodes. AS-nodes contain the same dependent probabilities between I-nodes, have tail I-nodes and a head I-node, and for any I-node in its head, only one S-node or AS-node can be active at any given time. The major difference is in functionality. Where S-nodes represent the known probabilities of a BKB, AS-nodes represent the unknown or missing probabilities of the BKB. By adding AS-nodes to a BKB, we are in fact, including the incomplete knowledge, or a close approximation of it, back into the BKB. By summing the probabilities of the S-nodes and AS-nodes to 1, and having the AS-nodes assume the highest value they can with regard to the Laws of Probability, we can achieve a tight upper bound as described in Definition 4.0.1. If we cannot achieve a summation to one, we must over-estimate the probabilities of the AS-nodes, and settle for an upper bound.

4.1 Assumptions

In order to achieve an upper bound using AS-nodes, we must make four necessary assumptions. Without these assumptions, AS-node creation becomes untenable.

1. PDs potentially responsible for a given BKB obey the Laws of Probability, i.e., probabilities always sum to 1.
2. The knowledge engineer who has created a BKB attempted to mirror one of these PDs. The normalization of a BKB's probabilities to maintain its consistency[2] will be deemed necessary to maintain mappings to one of these PDs. This enforces Constraint 8.
3. A r.v. A is dependent on another r.v. B if and only if there exists a directed path, via an S-node, from an I-node of B to an I-node of A. Not making this assumption implies a r.v. is potentially dependent on every other r.v. in the

BKB. This assumption prevents AS-node creation from exploding combinatorially in the worst case.

4. For each r.v., all of its I-nodes that could potentially impact the formation of probabilities with other r.v.s. are known, and are included in the BKB. Bayes' Theorem can only be used to fathom probabilities from known *dependencies.* Since we do not know how many (if any) I-nodes are missing, let alone what I-nodes in the BKB are dependent on them, working with missing I-nodes becomes intractable.
5. The BKB is consistent.

4.2 AS-node Creation Scenarios

To create an AS-node, we begin by examining each r.v.[1] in the BKB, but only one r.v. at a time, and determining what r.v.s. it is dependent on. Mutually exclusive (MUTEX) sets are then constructed by taking all combinations of I-nodes from these dependent r.v.s., but only one I-node from each r.v. will appear in any MUTEX set. For example, if we are dealing with r.v.s. with only two states, say TRUE and FALSE, there will be 2^n MUTEX sets containing the probabilities for each r.v. in the BKB, where n is the number of other r.v.s. that a r.v. is dependent on. Each MUTEX set can be thought of as a "bucket" that contains a sum of the probabilities for a r.v., and each "bucket" is labelled with a MUTEX combination of dependent I-nodes. Generally, a BKB will only contain a fraction of the necessary S-nodes to populate these MUTEX sets. By collecting the S-nodes that are present and placing them in the appropriate MUTEX set, we can take 1 minus the summation of their probabilities and determine the missing complement probability, if it is missing at all. An S-node is considered a member of a MUTEX set if its tail I-nodes form a subset of the MUTEX set's label.

[1]An r.v. is the union of its I-nodes.

The method by which an AS-node is created is based purely on the dependencies shared between r.v.s.. Since we know that all tail compatible probabilities must sum to no more than 1, we can use this fact along with the existing knowledge in the BKB to fathom what knowledge could be missing. To this end, we have developed five different scenarios for creating AS-nodes for a single r.v., considering only the other r.v.s. it is dependent on and the S-nodes involved between them. There are four important points to keep in mind when applying these scenarios:

1. Each r.v. in a BKB must be considered, and for any r.v., multiple scenarios can exist.
2. Each MUTEX set will have at most one scenario applied to it.
3. The scenario to be used is determined solely by the MUTEX set, and the known S-nodes that occupy that MUTEX set.
4. The results of one scenario do not immediately affect the results of any another scenario, thus each scenario is applied independently. It is possible that a conflict will arise where the probability of an existing S-node is changed multiple times because the S-node resides in more than one MUTEX set. In this case, all results for the *r.v.* should be cached and the conflict be appropriately resolved. This will be discussed in more detail in Section 4.3.

4.2.1 Scenario 1: Pure S-node Creation. For any MUTEX set, if there exists a single I-node whose probability is unknown, we can use the Laws of Probability to determine what the missing probability is. This is performed by summing the known probabilities for the other I-nodes and subtracting the total from 1.

For example, let's examine a BKB with instantiations consisting only of two values. The r.v. we are concentrating on is D, and it is dependent on A, B, and C. For the MUTEX set {A=a0, B=b0, C=c0}[2], we know that

$$P(D = d0|A = a0, B = b0, C = c0) = .70$$

[2]There are a total of $3^3 = 27$ MUTEX sets, but for this example, we will only examine this one.

the Laws of Probability tells us that

$$P(D = d1|A = a0, B = b0, C = c0) = .30$$

This knowledge can be added as a permanent S-node to the BKB. Similarly, in a BKB where r.v.s. have three I-nodes, S-nodes are constructed in a like manner when only one value of the MUTEX set is missing. Let the r.v.s. in this BKB have three I-nodes. If we are given

$$P(D = d0|A = a0, B = b0, C = c0) = .70$$
$$P(D = d1|A = a0, B = b0, C = c0) = .10$$

the Laws of Probability tells us that

$$P(D = d2|A = a0, B = b0, C = c0) = .20$$

This method generalizes for all cases where the summation of the probabilities is less than 1 and there exists a single missing probability.

4.2.2 Scenario 2: Summation Less Than One And Multiple Missing Probabilities. Let us continue using our BKB with three instances. If our MUTEX set still sums to less than 1 and we are missing more than one S-node, it is possible that any one of the remaining I-nodes could be supported by the difference of the probability. For example, given

$$P(D = d0|A = a0, B = b0, C = c0) = .70$$

both of the missing S-nodes for D=d1 and D=d2 could contain the remaining .3, in the best case. Two AS-nodes would be constructed to reflect

$$P(D = d1|A = a0, B = b0, C = c0) = .30$$
$$P(D = d2|A = a0, B = b0, C = c0) = .30$$

Creating both AS-nodes will result in the summation

$$.70 + .30 + .30 = 1.30 ¿ 1$$

a violation of Constraint 8. However, since only one of the created AS-nodes can be selected during inferencing, the MUTEX set essentially sums to 1.

$$.70 + (.30 \text{ XOR } .30) = 1$$

Although a consistency check will reveal the BKB to be inconsistent, for our purposes, we only care about the probability of the optimal solution. This probability will serve as an upper bound, or if possible, a tight upper bound. Therefore, a BKB with AS-nodes is allowed to be inconsistent.

4.2.3 Scenario 3: Summation Equal To One And Multiple Missing Probabilities. This scenario can occur because the remaining I-nodes either had S-nodes with probabilities of 0 so they were omitted, or the existing S-nodes were normalized for consistency purposes[2]. Normalization occurs when a set of probabilities that are not mutually exclusive sum to greater than 1. It then becomes necessary to normalize the probabilities so their sum equals 1 in order to maintain there relative frequency and also to keep the BKB consistent via Constraint 8[2]. The normalization process occurs during Knowledge Acquisition.

The first case is easy to verify. Simply maintain both the original and current (normalized) probabilities for each S-node in the knowledge base and check to see if they are equal. If they are equal, it implies the values were not normalized, and the knowledge engineer intended the probabilities to sum to 1. Therefore, no missing knowledge exists and nothing needs to be done. This first case will undoubtedly be very rare, since the knowledge engineer will infrequently plan for this to happen. If the original and current probabilities are different, we now consider the second case.

The second case is a little more difficult. Since the probabilities were normalized to 1, 1 is no longer a relative number. Normalization has skewed the original probabilities. For example, during Knowledge Acquisition if a MUTEX set contained probabilities

$$.90 + .90 + .90 + .90 = 3.60$$

it would become necessary to normalize them to

$$.25 + .25 + .25 + .25 = 1$$

Before the probabilities were normalized, the maximum probability that an S-node could assume is 1. Thus, to create an AS-node, we must also give it an initial probability of 1. The total number of AS-nodes to be created will equal the number of S-nodes missing from the MUTEX set. Since our goal is to achieve a summation to 1, the AS-nodes must be normalized in conjunction with the existing S-nodes. Therefore, normalize each AS-node and S-node using their ORIGINAL probabilities, that is, 1 for the AS-nodes and the original probabilities for the S-nodes. For example, in a BKB where each r.v. has five I-nodes, if

$$\mathrm{P(D = d2|A = a0, B = b0, C = c0) = .308, (original\ .40)}$$
$$\mathrm{P(D = d3|A = a0, B = b0, C = c0) = .154, (original\ .20)}$$
$$\mathrm{P(D = d4|A = a0, B = b0, C = c0) = .538, (original\ .70)}$$

our AS-nodes would contain

$$\mathrm{P(D = d0|A = a0, B = b0, C = c0) = 1}$$
$$\mathrm{P(D = d1|A = a0, B = b0, C = c0) = 1}$$

and after all probabilities are normalized, we would have

$$\mathrm{P(D = d0|A = a0, B = b0, C = c0) = .303}$$
$$\mathrm{P(D = d1|A = a0, B = b0, C = c0) = .303}$$
$$\mathrm{P(D = d2|A = a0, B = b0, C = c0) = .121}$$
$$\mathrm{P(D = d3|A = a0, B = b0, C = c0) = .061}$$
$$\mathrm{P(D = d4|A = a0, B = b0, C = c0) = .212}$$

This is the only method we could develop to handle both the missing knowledge, and the problem of requiring summation to 1.

4.2.4 Scenario 4: Summation Less Than One And No Missing Probabilities.

There are two explanations for this situation. The first implies the existence of another I-node(s) not included in the BKB. Since Assumption 4 states that all I-nodes that could potentially impact the formation of probabilities are included in the BKB, this first explanation is not a possibility. The second implies the probabilities were entered as such, and were not intended to sum to 1. In [2], probability ranges with tags were used to make knowledge acquisition easier for the knowledge engineer. They are shown below:

inconceivable:	0.00 - 0.10
not likely:	0.10 - 0.35
possible:	0.35 - 0.65
probable:	0.65 - 0.90
almost certain:	0.90 - 1.00

Only on rare occasions would probabilities sum to exactly 1. BKBs by definition do not require summation to 1, however, Assumptions 1, 2, and 4 do require summation to 1 in this special case. Therefore, we need to normalize the known probabilities and force them to sum to 1 in order to achieve an upper bound. For example, if a BKB has three instances and we know that

$$\mathrm{P(D = d0|A = a0, B = b0, C = c0) = .40}$$
$$\mathrm{P(D = d1|A = a0, B = b0, C = c0) = .20}$$
$$\mathrm{P(D = d2|A = a0, B = b0, C = c0) = .20}$$

normalizing these probabilities to sum to 1 would yield

$$\mathrm{P(D = d0|A = a0, B = b0, C = c0) = .50}$$
$$\mathrm{P(D = d1|A = a0, B = b0, C = c0) = .25}$$
$$\mathrm{P(D = d2|A = a0, B = b0, C = c0) = .25}$$

By normalizing probabilities, we maintain the ratio between them, and also guarantee an upper bound by summing them to 1.

A special case of this scenario is the NULL MUTEX set, which *only* occurs in root r.v.s.. Since the S-nodes of a root r.v. have no dependent tail I-nodes, all of the S-nodes will fill the NULL MUTEX set. In this case, we simply normalize all of the probabilities to ensure they sum to 1.

4.2.5 Scenario 5: Summation Equal To Zero. In the event that a MUTEX set has no existing S-nodes in it, the unassigned probability is 1. This is a special case of Scenario 2. AS-nodes with probability 1 would be created with tail I-nodes the same as those defining membership into the MUTEX set. Each AS-node would have a different head I-node covering all possible instances of the head r.v.. For example, if each r.v. has three I-nodes and r.v. D has no S-nodes covering the MUTEX set {A=a1, B=b1, C=c1}, then we would create the following AS-nodes:

$$\mathrm{P(D = d0|A = a1, B = b1, C = c1) = 1}$$
$$\mathrm{P(D = d1|A = a1, B = b1, C = c1) = 1}$$
$$\mathrm{P(D = d2|A = a1, B = b1, C = c1) = 1}$$

This type of AS-node, whose probability equals 1, will dominate all other AS-nodes when calculating an upper bound. Whenever the appropriate dependencies are available, this AS-node will be selected in preference to all others.

4.3 The AS-node Creation Process

As mentioned earlier, conflicts can arise when an S-node occupies more than one MUTEX set and one of those sets is normalized. If this occurs, the probability of that S-node is no longer constant, and it will directly affect the probabilities in the other MUTEX sets. To achieve a tight upper bound, each MUTEX set must sum to 1. The method in which we resolve these conflicts directly affects whether or not we can sum to 1, and thus attain a tight upper bound. There are three approaches to handling these conflicts:

1. Allow the probabilities to be changed in each MUTEX and create new S-nodes to hold these probabilities in their respective set. The problem with this approach is that in order for the S-node to be in more than one set, the number of I-nodes in the S-node's tail is less than the number I-nodes needed to "label" each MUTEX set. Creating S-nodes to hold the different probabilities

will alter the initial knowledge of the BKB, so the S-node must hold the same probability in each MUTEX set. Thus, this approach is not considered.

2. Normalizing a MUTEX set will send a ripple of changes to MUTEX sets that share its S-nodes. These changed sets must now normalize to sum to 1, but they will affect the MUTEX set that instigated the change. Thus, we must attempt to normalize all of the sets at once, to prevent this see-sawing of normalizations. If a MUTEX set must be normalized, find the MUTEX sets that share S-nodes with that set, and find the MUTEX sets that share S-nodes with these sets, and so on. By successfully applying Theorem 4.3.1, we can have each of the intersecting MUTEX sets sum to 1, and thus in part, guarantee that a tight upper bound is possible.
3. If applying Theorem 4.3.1 results in failure, the only recourse is to normalize the MUTEX sets independently, and use the highest probability for each S-node. Since the summations for each set no longer sum to 1, we can not guarantee a tight upper bound, however, an upper bound is still possible.

THEOREM 4.3.1. *A group of intersecting MUTEX sets can be successfully normalized to 1 if and only if the sum of each intersecting MUTEX sets' ORIGINAL probabilities are equal.*

Proof. Let MUTEX set $A = \{a_0 + a_1 + \ldots + a_m + s\}$ and let MUTEX set $B = \{b_0 + b_1 + \ldots + b_n + s\}$, where a_i, b_i, and s are S-node probabilities. Since both set A and B share S-node s, they must be normalized at the same time. Let the original sum of probabilities of set A be p_A, and the original sum of probabilities of set B be p_B.

When normalizing probabilities we divide each member of each set by the original sum of that set. However, S-node s can only hold one value, thus we know

$$\frac{s}{p_A} = \frac{s}{p_B}$$

$$s * p_A = s * p_B$$

$$p_A = p_B$$

Therefore, the sum of each intersecting MUTEX sets' original probabilities must be equal in order for the the MUTEX set to be properly normalized. Obviously, the proof can be reversed to prove the bi-conditional. □

The AS-node Creation Scenarios in Section 4.2 are the heart of entire AS-node Creation Process. They are the workhorse for altering existing probabilities, and adding new probabilities in the form of AS-nodes. To this point, we have only alluded to how these Scenarios would be utilized. We are now ready to present the formal AS-node Creation Algorithm.

```
FOR every r.v. $x$
    Create the MUTEX sets for $x$
    Fill each MUTEX set with the S-nodes of $x$

    FOR every MUTEX set
        Apply the appropriate Scenario
        Cache each new AS-node
        Cache the altered probability for each normalized S-node
    END FOR

    IF an S-node is normalized more than once
        Resolve the conflicts for $x$
        Create the cached AS-nodes for $x$
        Normalize the cached S-nodes for $x$
    END IF
END FOR
```

The algorithm's complexity is $O(rd)$, where r is the number of r.v.s., and d is the maximum product of MUTEX sets and S-nodes for any r.v. x in the BKB.

To illustrate the AS-node Creation Algorithm, we will apply it to the BKB in Figure 4.1. Examining the BKB, there are a total of six r.v.s., four of which are roots. We will process the roots first for clarity, however the relative order of the r.v.s. is normally unimportant. The r.v.s. A, C, and E all have probabilities that

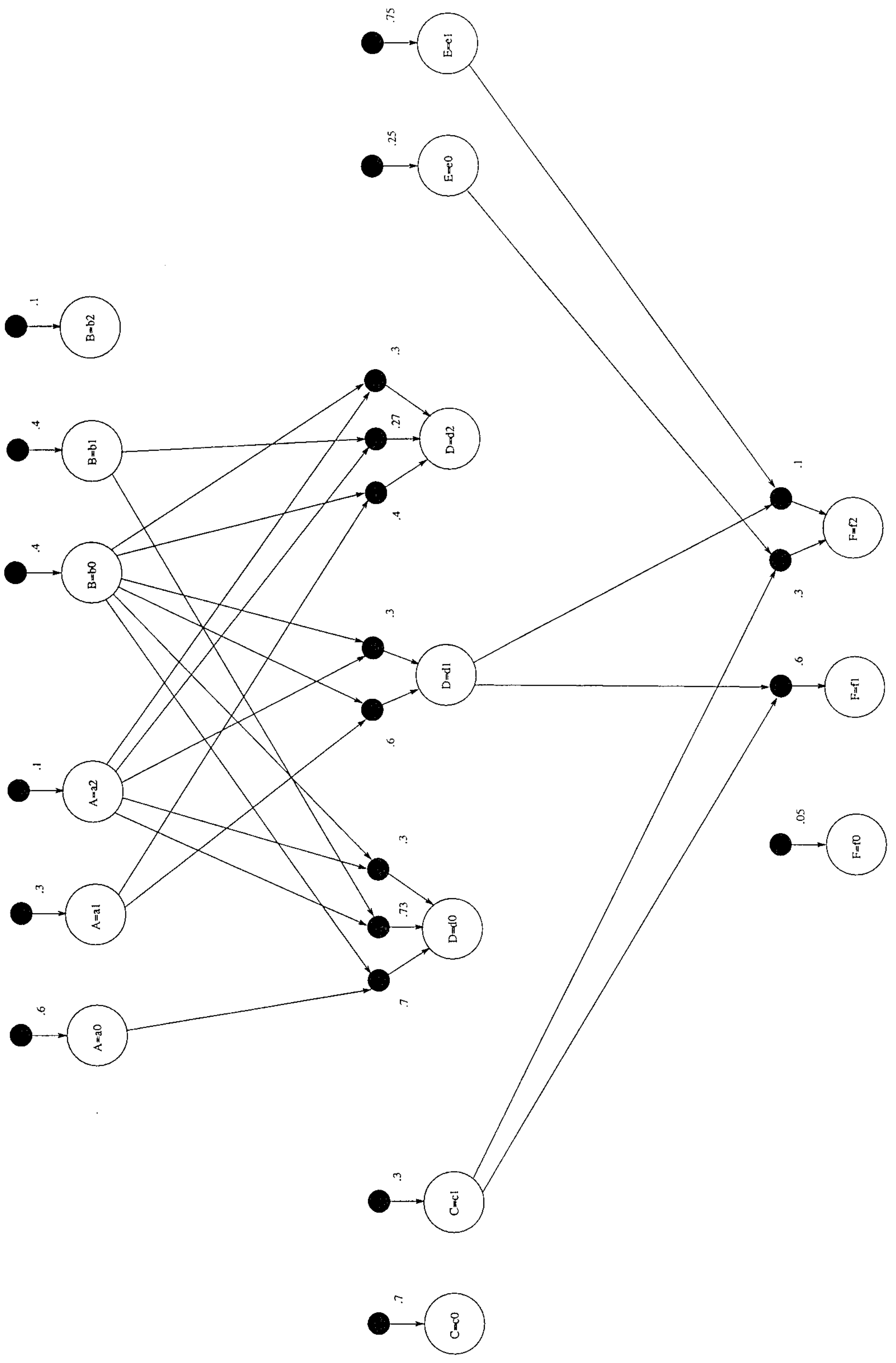


FIG. 4.1. A small Bayesian Knowledge Base

sum to 1, so nothing needs to be done. However, r.v. B only sums to .9, so we will use Scenario 4's special case and normalize their probabilities. The new probabilities are

$$P(B = b0) = .44$$
$$P(B = b1) = .44$$
$$P(B = b2) = .11$$

That only leaves two r.v.s., D and F. Consider r.v. D. D is dependent on two r.v.s., A and B, both of which have three I-nodes. Therefore, there are nine MUTEX sets. Filling the MUTEX sets with their appropriate probabilities yields:

$$\begin{array}{lllr}
1. & \{A = a0, B = b0\} \rightarrow & .70 & = 0.7 \\
2. & \{A = a0, B = b1\} \rightarrow & & = 0.0 \\
3. & \{A = a0, B = b2\} \rightarrow & & = 0.0 \\
4. & \{A = a1, B = b0\} \rightarrow & .60 + .40 & = 1.0 \\
5. & \{A = a1, B = b1\} \rightarrow & & = 0.0 \\
6. & \{A = a1, B = b2\} \rightarrow & & = 0.0 \\
7. & \{A = a2, B = b0\} \rightarrow & .30 + .30 + .30 & = 0.9 \\
8. & \{A = a2, B = b1\} \rightarrow & .27 + .73 & = 1.0 \\
9. & \{A = a2, B = b2\} \rightarrow & & = 0.0
\end{array}$$

Set 1 has a summation less than 1 and two missing probabilities, so we can use Scenario 2 to create AS-nodes

$$P(D = d1|A = a0, B = b0) = .30$$
$$P(D = d2|A = a0, B = b0) = .30$$

Set 4 has a summation equal to 1 and no missing probabilities. Upon further investigation, we discover that these are in fact the original probabilities, so we do nothing. Set 7 has a summation less than 1 and no missing probabilities, so we can use Scenario 4 to normalize the existing probabilities to

$$P(D = d0|A = a2, B = b0) = .33$$
$$P(D = d1|A = a2, B = b0) = .33$$

$$P(D = d2|A = a2, B = b0) = .33$$

Set 8 has a summation equal to 1 and no missing probabilities. Upon further investigation, we discover that the original probabilities were .8 and .3, and were normalized to their current values. Therefore, we can use Scenario 3 to create one AS-node and normalize the current probabilities to

$$P(D = d0|A = a2, B = b1) = .38$$
$$P(D = d1|A = a2, B = b1) = .48$$
$$P(D = d2|A = a2, B = b1) = .14$$

Sets 2, 3, 5, 6, and 9 all have summations equal to 0, so we can use Scenario 5 to create the following AS-nodes:

$$P(D = d0|A = a0, B = b1) = 1$$
$$P(D = d1|A = a0, B = b1) = 1$$
$$P(D = d2|A = a0, B = b1) = 1$$
$$P(D = d0|A = a0, B = b2) = 1$$
$$P(D = d1|A = a0, B = b2) = 1$$
$$P(D = d2|A = a0, B = b2) = 1$$
$$P(D = d0|A = a1, B = b1) = 1$$
$$P(D = d1|A = a1, B = b1) = 1$$
$$P(D = d2|A = a1, B = b1) = 1$$
$$P(D = d0|A = a1, B = b2) = 1$$
$$P(D = d1|A = a1, B = b2) = 1$$
$$P(D = d2|A = a1, B = b2) = 1$$
$$P(D = d0|A = a2, B = b2) = 1$$
$$P(D = d1|A = a2, B = b2) = 1$$
$$P(D = d2|A = a2, B = b2) = 1$$

We are finished with the AS-node Creation Process for r.v. D. Since there are no conflicts due to normalizing the same probability multiple times, all of the above changes will occur as is.

Continuing with r.v. F, F is dependent on three r.v.s., C, D, and E, each of which has two, three, and two I-nodes, respectively. Therefore, there are twelve MUTEX sets. Filling the MUTEX sets with their appropriate probabilities yields:

1. $\{C = c0, D = d0, E = e0\} \rightarrow .05 = .05$
2. $\{C = c0, D = d0, E = e1\} \rightarrow .05 = .05$
3. $\{C = c0, D = d1, E = e0\} \rightarrow .05 = .05$
4. $\{C = c0, D = d1, E = e1\} \rightarrow .05 + .10 = .15$
5. $\{C = c0, D = d2, E = e0\} \rightarrow .05 = .05$
6. $\{C = c0, D = d2, E = e1\} \rightarrow .05 = .05$
7. $\{C = c1, D = d0, E = e0\} \rightarrow .05 + .30 = .35$
8. $\{C = c1, D = d0, E = e1\} \rightarrow .05 = .05$
9. $\{C = c1, D = d1, E = e0\} \rightarrow .05 + .60 + .30 = .95$
10. $\{C = c1, D = d1, E = e1\} \rightarrow .05 + .60 + .10 = .75$
11. $\{C = c1, D = d2, E = e0\} \rightarrow .05 + .30 = .35$
12. $\{C = c1, D = d2, E = e1\} \rightarrow .05 = .05$

Sets 1, 2, 3, 5, 6, 8, and 12 all have a summation less than 1 and two missing probabilities, so we can use Scenario 2 to create AS-nodes

$$P(F = f1|C = c0, D = d0, E = e0) = .95$$
$$P(F = f2|C = c0, D = d0, E = e0) = .95$$
$$P(F = f1|C = c0, D = d0, E = e1) = .95$$
$$P(F = f2|C = c0, D = d0, E = e1) = .95$$
$$P(F = f1|C = c0, D = d1, E = e0) = .95$$
$$P(F = f2|C = c0, D = d1, E = e0) = .95$$
$$P(F = f1|C = c0, D = d2, E = e0) = .95$$
$$P(F = f2|C = c0, D = d2, E = e0) = .95$$
$$P(F = f1|C = c0, D = d2, E = e1) = .95$$
$$P(F = f2|C = c0, D = d2, E = e1) = .95$$
$$P(F = f1|C = c1, D = d0, E = e1) = .95$$

$$P(F = f2|C = c1, D = d0, E = e1) = .95$$
$$P(F = f1|C = c1, D = d2, E = e1) = .95$$
$$P(F = f2|C = c1, D = d2, E = e1) = .95$$

Sets 4, 7, and 11 all have a summation less than 1 and a single missing probability. Scenario 1 can create the following AS-nodes which can be permanently added to the BKB

$$P(F = f1|C = c0, D = d1, E = e1) = .85$$
$$P(F = f1|C = c1, D = d0, E = e0) = .65$$
$$P(F = f1|C = c1, D = d2, E = e0) = .65$$

Set 9 has a summation less than 1 and no missing probabilities, so we can use Scenario 4 to normalize the existing probabilities to

$$\begin{aligned} P(F = f0) &= .05 \\ P(F = f1|C = c1, D = d1) &= .63 \\ P(F = f2|C = c1, E = e0) &= .32 \end{aligned}$$

Similarly, Set 10 also has a summation less than 1 and no missing probabilities, so we can use Scenario 4 to normalize the existing probabilities to

$$\begin{aligned} P(F = f0) &= .07 \\ P(F = f1|C = c1, D = d1) &= .80 \\ P(F = f2|D = d1, E = e1) &= .13 \end{aligned}$$

Notice that AS-nodes

$$\begin{aligned} P(F = f0) &= .05 \\ P(F = f1|C = c1, D = d1) &= .60 \end{aligned}$$

are normalized more than once. In fact, they appear in more than one MUTEX set. Theorem 4.3.1 can not be successfully applied to at least one of the conflicting MUTEX sets, thus we will have to resort to our third method for handling conflicts. Since we initially considered each scenario independently, we did not use their changed values in the other scenarios. Furthermore, since we are finished with this r.v., and a conflict exists for AS-node

$$\begin{aligned} P(F = f0) &= .05 \\ P(F = f1 | C = c1, D = d1) &= .60 \end{aligned}$$

we will normalize them to .07 and .80 respectively, their highest probabilities during the AS-node Creation Process. Since we could not resolve the conflicts using Theorem 4.3.1, inferences to the BKB will only produce an upper bound. This concludes our example for the BKB in Figure 4.1.

4.4 AS-node Reduction Techniques

The AS-node Creation Process is for the most part, a brute force method for computing the possible probabilities for a given r.v., however, there seems to be little alternative. Regardless of the number of existing S-nodes, we will always produce a combinatoric number of AS-nodes. Therefore, reducing the number of AS-nodes whenever possible is very important. Since AS-nodes reproduce the combinatorics inherent in Bayesian Networks[23], inference run times for a BKB with AS-nodes will be much slower than for those without. Section 4.4.1 and Section 4.4.2 are both methods for reducing the number of AS-nodes added to a BKB, however Section 4.4.1 maintains a tight upper bound. Section 4.4.2, while only guaranteeing an upper bound, does provide a much faster inference time especially for General and Cyclic BKBs. The AS-node property for providing an upper bound, and possibly a tight upper bound, is summarized in Theorem 4.4.1.

THEOREM 4.4.1. *Applying the AS-node Creation Algorithm to a BKB provides an upper bound to any query to the original BKB. Successfully applying both the AS-node Creation Algorithm and Theorem 4.3.1 to a BKB provides a tight upper bound to any query to the original BKB.*

Proof. To prove Theorem 4.4.1, the key is to guarantee that any I-node or AS-node in the BKB can assume it's maximum probability for any possible combination of dependent I-nodes, and also that each MUTEX set sums to 1. The five scenarios in

Section 4.2 outline all possible cases. Thus, by proving these scenarios and resolving conflicts appropriately, we prove Theorem 4.4.1.

For each scenario, assume we are creating AS-nodes for a single r.v. without loss of generality. Let $v_1, v_2, \ldots, v_k$ be the known I-node probabilities for a dependent MUTEX set and $p_1, p_2, \ldots, p_n$ be the unknown I-node probabilities for a dependent MUTEX set. Also, $0 \leq v_i, p_i \leq 1$.

Scenario 1: The Laws of Probability tell us that all tail compatible probabilities must sum to 1, therefore

$$v_1 + v_2 + \ldots + v_k + p_1 = 1$$

$$p_1 = 1 - v_1 + v_2 + \ldots + v_k$$

Scenario 2: Following from scenario 1, we know the remaining unknown probabilities must sum to 1 and $p_i = 1 - v_1 + v_2 + \ldots + v_k$ so,

$$v_1 + v_2 + \ldots + v_k + p_1 + p_2 + \ldots + p_n \neq 1$$

but since at most one I-node can be selected for any single r.v.,

$$v_1 + v_2 + \ldots + v_k + (p_1 \text{ XOR } p_2 \text{ XOR } \ldots \text{ XOR } p_n) = 1$$

This will satisfy the Laws of Probability and also yield an upper bound.

Scenario 3: In the case where normalization must take place, scenario 3 dictates,

$$\frac{1}{v_1 + v_2 + \ldots + v_k + 1} \geq \frac{p_1}{v_1 + v_2 + \ldots + v_k + p_1}$$

$$\begin{aligned}
(1)(v_1 + v_2 + \ldots + v_k + p_1) &\geq p_1(v_1 + v_2 + \ldots + v_k + 1) \\
v_1 + v_2 + \ldots + v_k + p_1 &\geq v_1p_1 + v_2p_1 + \ldots + v_kp_1 + p_1 \\
v_1 + v_2 + \ldots + v_k &\geq v_1p_1 + v_2p_1 + \ldots + v_kp_1 \\
v_1 + v_2 + \ldots + v_k &\geq p_1(v_1 + v_2 + \ldots + v_k) \\
v_1 + v_2 + \ldots + v_k &\geq p_1(v_1 + v_2 + \ldots + v_k) \\
1 &\geq p_1
\end{aligned}$$

p_1 must assume a probability greaten than 1 in order to exceed the probability of the AS-node, a clear violation of the Laws of Probability. Therefore, the AS-node will bound the probability p_1.

Scenario 4: We know that $v_1 + v_2 + \ldots + v_k \leq 1$, and there exists no probabilities p_i. Satisfying the Laws of Probability requires $v_1 + v_2 + \ldots + v_k = 1$ while maintaining the proportionality between the probabilities. This can be achieved by normalizing the probabilities.

Scenario 5: This is a special case of scenario 2 where $v_1 = v_2 = \ldots = v_k = 0$.

Since the algorithm independently caches the results of each scenario, we guarantee that the results of one scenario will not initially corrupt the results of another. If Theorem 4.3.1 is successfully applied at every r.v. in the BKB, every MUTEX set will sum to 1. This coupled with the fact that the AS-nodes created by Scenarios 1-5 will account for all missing probabilities, the BKB with AS-nodes will deliver a tight upper bound. However, if Theorem 4.3.1 is not successfully applied, we only save the maximum normalized probability for any single S-node. The AS-nodes will still account for all missing probabilities, but each MUTEX set will now sum to a value greater than or equal to 1. Thus, only an upper bound will be guaranteed. $\square$

4.4.1 Merging AS-nodes in a BKB. When creating AS-nodes in Scenarios 2, 3, and 5, we will generally be creating multiple AS-nodes. Since the tails of each AS-node are identical, we can merge the AS-nodes into a single node. For example in Section 4.2.2, we ended up creating two AS-nodes:

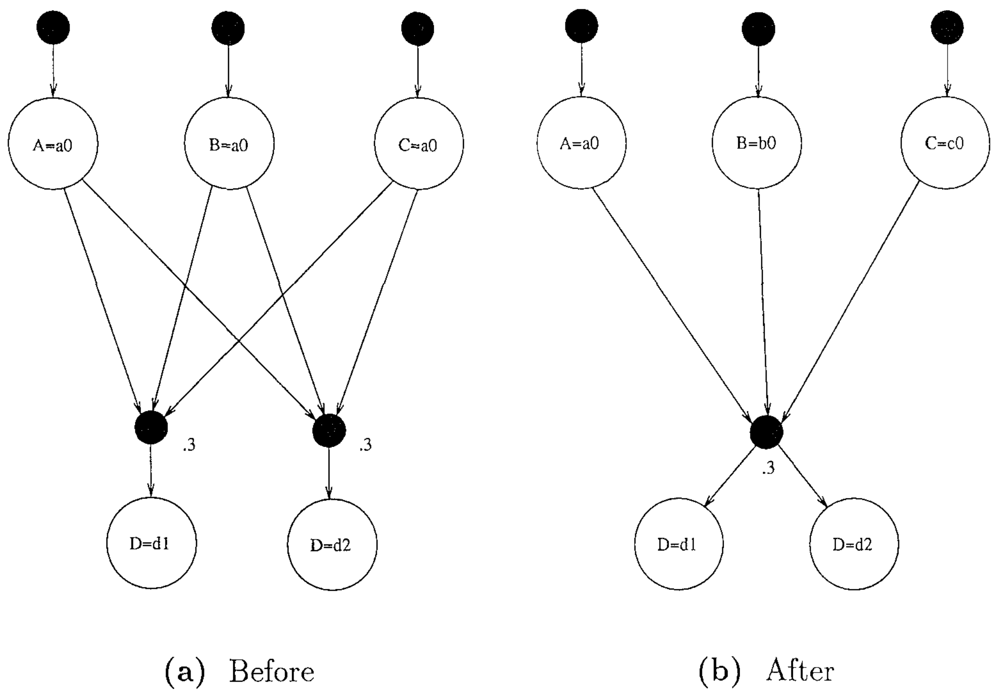


FIG. 4.2. Example of merging AS-nodes

$$P(D = d1 | A = a0, B = b0, C = c0) = .30$$
$$P(D = d2 | A = a0, B = b0, C = c0) = .30$$

Graphically, those AS-nodes would appear as Figure 4.2(a). Merging these two AS-node together would involve maintaining the tail dependencies on a single AS-node and then merging all of the head I-node dependencies. The resulting AS-node can be viewed in Figure 4.2(b). This operation can only be performed on AS-nodes with identical tails, identical probabilities, and head I-nodes belonging to the same r.v.. Since the necessary AS-nodes are always available at the same time during the AS-node Creation Process, this method is $O(1)$.

Additional merging can be performed on the AS-nodes generated in Scenario 5. Continuing our example from Figure 4.2(b), Scenario 5 would require us to create an additional $3^3 - 1 = 26$ AS-nodes with probability 1 to cover the other MUTEX sets, assuming we have already merged the head I-nodes. It is at this point that the savings can occur. Since r.v. D is not dependent on B=b1 after running Scenarios

1-4 (no associated S-nodes), B=b1 can be considered independent of r.v.s. A and C. The same can be said for B=b2 with respect to r.v.s. A and C, and A=a1 and A=a2 with respect to r.v.s. B and C, and also C=c1 and C=c2 with respect to r.v.s. A and B. Thus, we would only need to include six AS-nodes, one for each unused tail instance:

AS-node #1	**AS-node #2**	**AS-node #3**
$P(D = d0 \mid A = a1) = 1$	$P(D = d0 \mid A = a2) = 1$	$P(D = d0 \mid B = b1) = 1$
$P(D = d1 \mid A = a1) = 1$	$P(D = d1 \mid A = a2) = 1$	$P(D = d1 \mid B = b1) = 1$
$P(D = d2 \mid A = a1) = 1$	$P(D = d2 \mid A = a2) = 1$	$P(D = d2 \mid B = b1) = 1$

AS-node #4	**AS-node #5**	**AS-node #6**
$P(D = d0 \mid B = b2) = 1$	$P(D = d0 \mid C = c1) = 1$	$P(D = d0 \mid C = c2) = 1$
$P(D = d1 \mid B = b2) = 1$	$P(D = d1 \mid C = c1) = 1$	$P(D = d1 \mid C = c2) = 1$
$P(D = d2 \mid B = b2) = 1$	$P(D = d2 \mid C = c1) = 1$	$P(D = d2 \mid C = c2) = 1$

This method would also reduce the number of MUTEX sets required for Scenarios 1-4 if applied beforehand. In this example, there are 3 * 3 * 3 = 27 MUTEX sets for r.v. C. Since I-nodes A=a1, A=a2, B=b1, and B=b2 are independent, we no longer need to include them when calculating the number of MUTEX sets. We now only deal with (3 - 2) * (3 - 2) * (3 -2) = 1 MUTEX set, the one containing the existing S-node.

Identifying the AS-nodes for the preprocessing method can be performed in $O(n)$, where n is the number of S-nodes belonging to a given r.v.. Simply examine the tail I-nodes of every S-node belonging to the r.v. in question. Any dependent I-node not found in an S-node's tail is a *possible* candidate for this method. If the number of I-nodes in an S-node's tail is less than the number dependent r.v.s., then remove all of the missing r.v.'s I-nodes as candidates. Once AS-nodes are created using the remaining I-node candidates, the I-node candidates are ignored when computing the number of MUTEX sets. BKBs that are both sparse and contain r.v.s. with a large number of instances would benefit the greatest from this reduction method.

By reducing the total number of required AS-nodes, the overall size of the BKB is decreased while providing a net reduction in inferencing run times. In addition, both methods maintain a tight upper bound as stated in Theorem 4.4.2.

THEOREM 4.4.2. *Merging AS-nodes in a BKB maintains the tight upper bound property.*

Proof.

1. AS-nodes with identical tail probabilities and dependencies.
   Merging AS-nodes with identical tail probabilities and dependencies does not alter the probabilities nor dependencies of any I-node of that r.v.. Since only a single S-node or AS-node can be selected for any r.v., an AS-node with multiple head I-nodes will be selected by at most one I-node of that r.v..

2. AS-nodes with probabilities of 1.0.
   Let $V_1, V_2, \ldots, V_n$ be r.v.s. with instances denoted $v1_1, v1_2, \ldots, v1_m$ for V_1 and similarly for the others. Let V_n be dependent on $V_1, V_2, \ldots, V_{n-1}$.
   If $P(V_n|V_1 = v1_1, V_2, V_3, \ldots, V_{n-1}) = 1$, then we know that when $V_1 = v1_1$, $P(V_n) = 1$ regardless of what instance $V_2, V_3, \ldots, V_{n-1}$ assumes. Therefore, V_n is independent of $V_2, V_3, \ldots, V_{n-1}$, but only when $V_1 = v1_1$, thus $P(V_n|V_1 = v1_1, V_2, V_3, \ldots, V_{n-1}) = P(V_n|V_1 = v1_1) = 1$. This argument generalizes for all instances of $V_2, V_3, \ldots, V_{n-1}$.

Since the joint probability of the r.v. is not changed, the tight upper bound property is maintained. □

4.4.2 Cutting AS-nodes from a BKB. In large BKBs, the AS-node Creation Process reproduces the combinatorics inherent in a Bayesian Network. As such, merging AS-nodes alone may not sufficiently decrease the total number of AS-nodes to bring inferencing run times to a reasonable interval. The number of AS-nodes can be significantly reduced by cutting the number of AS-nodes generated by Scenarios 1-4 down to a single AS-node per I-node. This can result in a material decrease in the number AS-nodes added to a BKB, and therefore potentially reduce inferencing times[8, 16] to an acceptable range as determined by the knowledge engineer. This method no longer preserves the tight upper bound property, however, it does maintain an upper bound via Theorem 4.4.3. In some instances, an upper bound may suffice, so by trading-off accuracy, we can gain the necessary inferencing speed. Whether or not to use this reduction is totally subject to the domain in question, and the needs of the knowledge engineer.

The AS-node cutting method is quite straightforward. AS-nodes which do not hold a probability of 1 are the only nodes to be considered. For each I-node, eliminate every AS-node except the one with the largest probability, and also eliminate all of the AS-node's tail dependencies. This allows the I-node to assume the highest probability all of its AS-nodes would have allowed regardless of dependencies. This cutting will maintain the upper bound property, but the loss of the dependencies will no longer keep it tight.

The reason the AS-nodes with probability 1 were not considered is for a very obvious reason. For sparse BKBs (BKBs with very few S-nodes), it is conceivable that nearly every I-node will have an AS-node with probability 1. Any inference over a BKB that has cut away all of its AS-nodes, leaving only those with probability 1, would yield a probability at or very near to 1. This upper bound is very loose, to the extreme, and is of little use to the knowledge engineer, thus it is avoided.

THEOREM 4.4.3. *Cutting AS-nodes from a BKB no longer guarantees that the upper bound is tight.*

Proof. Applying Theorem 4.4.1 to a BKB yields $P(V|\pi_i) = p_i$ for any instance V dependent on the set of I-node combinations π with probability p, where each combination π_i corresponds to each p_i.

Let's examine those AS-node probabilities where $p_i < 1$.

$$P(V|\pi_i) = p_i$$

can be reduced to

$$P(V|\pi_i) = \max (p_i),$$

but this no longer ensures a tight upper bound, however, it does maintain an upper bound. This can be further reduced to

$$P(V) = \max (p_i)$$

This still maintains an upper bound since V can still only assume max (p_i), the maximum probability the AS-nodes would have provided. □

4.5 Inferencing with AS-nodes

Examining the effects of including AS-nodes in a BKB is a worthwhile venture, mainly to ascertain the appropriate inference engine to be used. The structure of the resulting BKB is solely determined by whether of not the original BKB contained cyclic knowledge.

If a BKB contains acyclic knowledge, the addition of AS-nodes is guaranteed to leave the BKB acyclic. This is due to the fact that the flow of knowledge is unidirectional, i.e., all of the BKB's "paths" are travelling in the same direction, and since AS-nodes include additional "paths" to the BKB based solely on the existing knowledge, they simply reinforce the unidirectional S-node "paths" that previously existed. Any of the inference engines in Chapter 3 can be used, provided the BKB in question has the appropriate structure. Using the inference engine for Cyclic BKBs, however, would be overkill.

On the other hand, the addition of AS-nodes to a BKB with cyclic knowledge will render the BKB cyclic. This can be readily seen by examining the three types of cyclic knowledge:

1. Type I cyclic knowledge is of the general form:

$$P(A = a0|B = b0) = .60 \tag{4.1}$$

$$P(B = b1|A = a1) = .25 \tag{4.2}$$

   Since the BKB is at the I-node level, the Type I paths never "intersect", and thus, never form a "true" cycle.

2. Type II cyclic knowledge is of the general form:

$$P(A = a0|B = b0, C = c0) = .60 \tag{4.3}$$

$$P(B = b0|A = a0, C = c1) = .25 \tag{4.4}$$

   Type II embodies the MUTEX cycles described in Constraint 5. A cycle is prevented from occurring in the solution subgraph due to the existence of r.v. C, the MUTEX variable, which forces only one path to be selected during inferencing.

3. Type III cyclic knowledge is of the general form:

$$P(A = a0|B = b0) = .60 \tag{4.5}$$

$$P(B = b0|A = a0) = .25 \tag{4.6}$$

   These are the cycles prevented by Constraint 5.

Normally, BKBs containing Type I and II cyclic knowledge would use the General BKB Inference Engine, but BKBs containing Type III cyclic knowledge would have no choice other than to use the Cyclic BKB Inference Engine.

The AS-node Creation Process can be employed in the same manner on BKBs containing all three types of cyclic knowledge, however, we will only consider BKBs with Type I and Type II since they do not already imply a Cyclic BKB. For example, take Equation 4.1 and Equation 4.2, and assume there is a third I-node for both A and B. We would include two additional AS-nodes producing:

$$P(A = a0|B = b0) = .60 \tag{4.7}$$

$$P(B = b0|A = a0) = .75 \tag{4.8}$$

$$P(B = b1|A = a1) = .25 \tag{4.9}$$

$$P(A = a1|B = b1) = .40 \tag{4.10}$$

However, these probabilities no longer represent Type I cyclic knowledge. In particular, Equation 4.7 and Equation 4.8, and Equation 4.9 and Equation 4.10 form cycles that now violate the now defunct Constraint 5. They are in fact, Type III cyclic knowledge. Therefore, any BKB containing cyclic knowledge that includes AS-nodes will become cyclic, and should use the Cyclic BKB Inference Engine to compute an upper bound.

4.6 The Utility of AS-nodes

AS-nodes are of benefit to the knowledge engineer. With numerous probabilities and instances to track, Knowledge Acquisition is a difficult task, however, Validating & Verifying the knowledge can be equally as trying. Every tool can help, and AS-nodes are just such a tool.

The first use of AS-nodes is summarized in Theorem 4.4.1. Too often, the meaning of the JPD is lost, and the incompleteness of the knowledge stymies the meaning even further. In a Bayesian Network, we know exactly what the optimal solution is, and also its probability. In a BKB, we do not have that luxury. Thus, AS-nodes can provide a measure of quality to a query by simply taking the difference between a query to a BKB with AS-nodes and the same query to the BKB without AS-nodes.

AS-nodes also provide a measure of the incompleteness of the BKB. This can be accomplished by:

1. Physically counting the number of AS-nodes generated, thus giving an exact number of the missing probabilities. This is only useful if the count is performed before any AS-node pruning techniques are employed. OR,
2. Calculating the JPD with no evidence. A high probability would indicate a large number of AS-nodes with probabilities of or near 1. This method can also be used to measure incompleteness in various regions of the BKB by simply pruning the BKB of all nodes except the region of interest.

Lastly, AS-nodes add new directed "paths" to the BKB. In the event that knowledge engineer does not receive a solution for a given set of evidence, AS-nodes can help identify the missing knowledge that is necessary for a query to succeed. He/she can perform this simply by instructing the inference engine to add the AS-nodes to the BKB, then clamping both the evidence and the expected solution. The inference engine will return the most probable path between them. Thus, the knowledge engineer can evaluate this path as a possible solution, or clamp additional r.v.s. until an expected path is returned. He can then add the necessary knowledge to make this path the returned solution.

V. Test Results

5.1 BKB Inference Engines

Both the BKB Polytree and General BKB inference engines were implemented during the course of this research. The BKB Polytree engine included none of the performance improvements from Section 3.4. However, the General BKB engine used Pruning Rules 1 and 2, and only selected I-node variables in the Branch and Bound tree. In addition, the Simplex method made use of sparse matrices to improve computation speed.

Thirty consistent Polytree BKBs were constructed consisting of 20, 30, and 50 r.v.s., 10 BKBs each. A r.v. had 2-4 I-nodes, and each I-node had 1-3 S-nodes. The BKBs utilized a layered structure such that the r.v.s. were evenly divided among four layers and dependencies only occurred between adjacent layers. The layered structure was chosen because of the similarity to the medical diagnosis decision trees used in [13]. The statistics for the 30 BKB test cases are presented in Table 5.1, where the term *nodes* refers to the total number of S-nodes and I-nodes in each BKB.

All 30 test cases were run with no evidence on both the BKB Polytree and General BKB inference engines. Computations were done on Sun Sparc 20 Workstations while keeping track of overall CPU usage. The results are plotted in Figures 5.1 and 5.2. As can be seen, the General BKB inference engine clearly outperformed the BKB Polytree inference engine, even on the 50 r.v. BKB Polytrees. Additional tests runs with larger BKBs should be performed to more fully quantify the run times of

	Min Nodes	Min Nodes	Avg Nodes
20 r.v.s.	131	145	138
30 r.v.s.	191	230	209
50 r.v.s.	317	356	337

TABLE 5.1. BKB Polytree statistics.

the General BKB inference engine. Even though number of nodes was used to compare the two inference engines, an alternate, and perhaps better, measure of BKB size for the General BKB inference engine is to use the number of linear constraints.

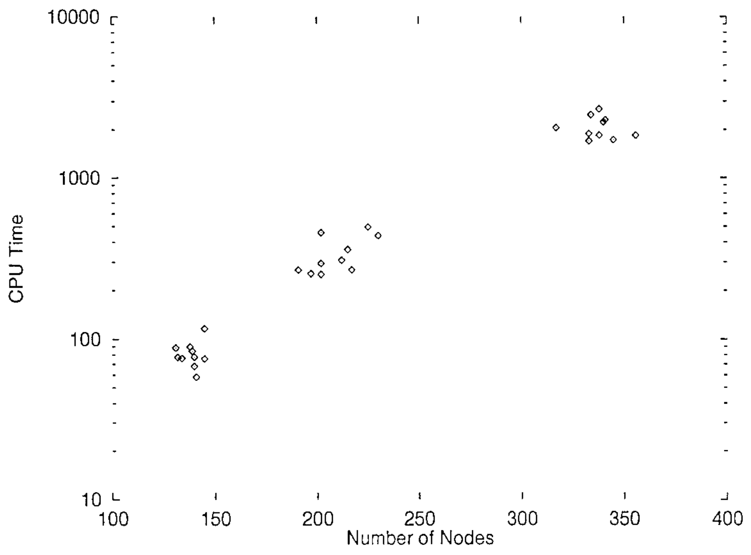


FIG. 5.1. BKB Polytree inference engine test results.

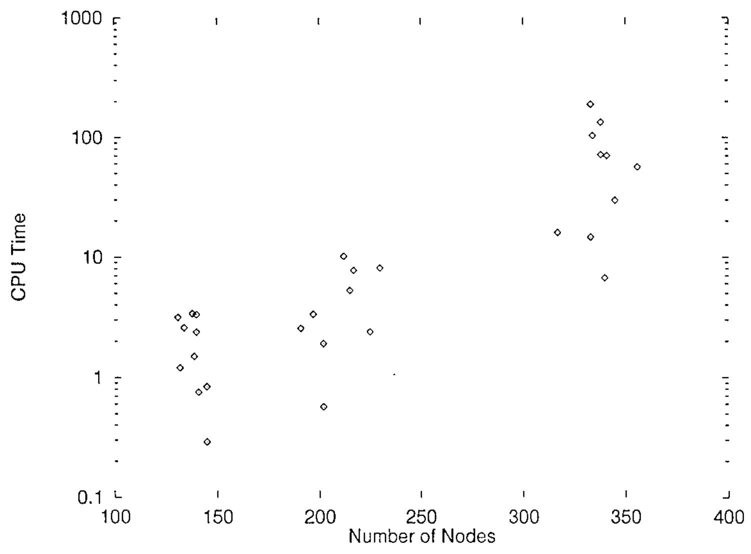


FIG. 5.2. General BKB inference engine test results.

	Original	With AS-nodes
BKB 1	4.48657e-10	3.398470e-04
BKB 2	1.42536e-09	7.247120e-04
BKB 3	1.03678e-09	9.311250e-04
BKB 4	2.61670e-07	4.672080e-05
BKB 5	1.40797e-09	1.117870e-03
BKB 6	9.38309e-09	4.269590e-04
BKB 7	1.94161e-08	4.780170e-04
BKB 8	4.13341e-08	8.195600e-05
BKB 9	3.42332e-11	2.473160e-04
BKB 10	4.03598e-10	1.621950e-04

TABLE 5.2. AS-node query results.

5.2 AS-nodes as an Organization & Validation Inferencing Tool

The 10 BKB Polytree test cases consisting of 20 r.v.s. from Section 5.1 were used as base cases for this experiment. AS-nodes were added to each of the BKBs using only merging techniques. One query was issued to each of the 10 BKB base cases, and an identical query issued to the corresponding BKB with AS-nodes. A single I-node was submitted as evidence for each query. The results are shown in Table 5.2. Of the 10 queries, the query to BKB 4 is of the highest quality[1], and the query to BKB 9, the lowest.

[1]Quality is measured by the difference in probabilities. A high difference implies a low quality, and vice versa.

VI. Conclusion

This research presents three inference engines that deliver optimal solutions by exploiting the structure of three distinct BKB subclasses. The first is the BKB Polytree engine, which uses a message passing scheme to compute the optimal solution. Its main advantage is its guaranteed worst case polynomial run time of $O(ln)$, but only on singly-connected BKBs. The final two inference engines utilize the Simplex method with Branch and Bound techniques to calculate the optimal solution. The General BKB engine can be used on multiply-connected BKBs that conform to the structure dictated by the Consistency Constraints purposed by Banks[2]. These constraints allow the inclusion of MUTEX cycles, which this inference engine can easily handle. It's run times surpassed those of the BKB Polytree engine when a solution was available, however when a solution is unavailable, the inference engine runs in exponential time while exhausting the entire solution space. The final inference engine is merely a modification to the General BKB inference engine. By including additional constraints to the inference engine, we can force a topological ordering of the I-nodes, and thus eliminate the selection of a cycle in the solution subgraph. As such, these additional constraints give the inference engine the ability to reason over virtually any BKB, most notably cyclic BKBs. However, the addition of these constraints can severely limit the size of the BKB to be inferenced. It also runs in exponential time when no solution exists.

Several performance improvement techniques for increased inferencing efficiency were discussed, the majority of which focussed on reducing the number of nodes within the BKB. Pruning rules allow the BKB to be reduced as a preprocessing method before an inference engine is evoked, and also at each node during the Branch and Bound procedure for the ILP-based inference engines. In addition, eliminating the need to include S-node variables in the Branch and Bound tree significantly reduced the ILP search space, thereby decreasing inference run times especially in the worse case. Finally, a hybrid method that combines the BKB Polytree inference engine with the ILP-based inference engines was presented. By ordering the I-node variables to the Branch and Bound tree, the BKB could be forced to become a BKB Polytree, eliminating the need to run the Simplex method at that node. This method produces an integral solution much quicker, whereas the Simplex method could continue deeper into the tree while arriving at non-integral solutions. In addition, jumpstart techniques were discussed to provide an initial solution to aid in constraining the search space by providing an initial bounding condition.

AS-nodes were presented as a inferencing tool to aid the knowledge engineer in BKB Organization & Validation. Since BKBs inherently deal with incomplete probabilities, AS-nodes deliver an upper bound, or possibly a tight upper bound, to the solution of any query, giving the solution a measure of quality when no absolute is available. In addition to measuring quality, AS-nodes can also measure incompleteness. This can be achieved by counting the number of AS-nodes added to a BKB, or simply by examining the probability of a query with no evidence. The more AS-nodes and the higher the probability, the more incomplete the BKB. Lastly, AS-nodes can propose possible solutions by adding paths to a BKB when no path exists between evidence and a desired solution. This could be of considerable benefit when knowledge grows too large during knowledge acquisition, and thus becomes unmanageable.

Further research by conducting empirical studies on our inference engines and performance improvement techniques is warranted, both in terms of improving inferencing speed, and also in terms of maximizing BKB size[1]. Another area to be explored is improving the ILP methods beyond our preprocessing and variable reduction methods. The improvement techniques presented in this paper focus mainly on reducing branching in the Branch and Bound procedure. Research into improved bounding techniques, either in the form of a jumpstart or otherwise, also needs to

[1]Size refers to both number of edges, r.v.s., I-nodes, and S-nodes.

be considered, in addition to a possible successor to the Simplex method. Almost no research has been conducted in these "Operations Research" areas and considerable gains can be made.

Bibliography

1. Baenen, Eric P. *Generalized Probabilistic Reasoning and Empirical Studies on Computational Efficiency and Scalability.* MS thesis, AFIT/GCE/ENG/94D-02. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1994.
2. Banks, Darwyn O. *Acquiring Consistent Knowledge for Bayesian Forests.* MS thesis, AFIT/GCE/ENG/94M-01. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, March 1995.
3. Bickmore, Timothy W. *SSME HPOTP Post-Test Diagnostic System Enhancement Project.* Final Report Contract NAS3-25883, Sacramento, California: Aerojet Propulsion Division, 2 February 1994.
4. Buchanan, Bruce G. and Edward H. Shortliffe. *Rule-Based Expert Systems.* Addison Wesley, 1984.
5. Charniak, Eugene and Solomon E. Shimony. "Probabilistic Semantics for Cost Based Abduction." *Proceedings of the AAAI Conference.* 106–111. 1990.
6. Cooper, Gregory F. "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks," *Artificial Intelligence*, *42*:393–405 (1990).
7. Cormen, Thomas H., Charles E. Leiserson and Ronald L Rivest. *Introduction to Algorithms.* McGraw-Hill Book Company, 1992.
8. Crowder, Harlan, Ellis L. Johnson and Manfred W. Padberg. "Solving Large-Scale Zero-One Linear Programming Problems," *Operations Research*, *31*:803–834 (1983).
9. Duda, Richard O., J. Gaschnig and P. Hart. "Model Design in the PROSPECTOR Consultant System for Mineral Exploration." *Expert Systems in the Micro-Electronic Age* 153–167, Edinburgh University Press, 1979.
10. Feigenbaum, Edward A., "Knowledge Engineering in the 1980's." Dept. of Computer Science, Stanford University, Stanford CA, 1982.
11. Garey, Michael R. and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and Company, 1979.
12. Geiger, Dan and David Heckerman. "Advances in Probabilistic Reasoning." *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence.* 118–126. 1991.
13. Healey, Patrice M. and Edwin J. Jacobson. *Common Medical Diagnoses: An Algorithmic Approach.* W. B. Saunders Company, 1990.
14. Hillier, Frederick S. and Gerald J. Lieberman. *Introduction to Operations Research.* Oakland, California: Holden-Day Inc., 1990.
15. Hobbs, Jerry R., Mark Stickel, Paul Martin and Douglas Edwards. "Interpretation as Abduction." *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics.* 95–103. 1988.

16. Hoffman, Karla L. and Manfred Padberg. "Improving LP-Representations of Zero-One Linear Programs for Branch-and-Cut," *ORSA Journal on Computing*, *3*(2):121–134 (1991).
17. Land, A. H. and A G. Doig. "An Automatic Method for Solving Discrete Programming Problems," *Econometrica*, *28*(3):497–520 (1960).
18. Li, Zhaoyu and Bruce D'Ambrosio. "A Framework for Ordering Composite Beliefs in Belief Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, *25*(2):243–255 (February 1995).
19. McCammon, Richard B. "Prospector II." *Proceedings of the Annual AI Systems in Government Conference*. 88–92. 1989.
20. Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1994.
21. Ng, Keung-Chi and Bruce Abramson. "Uncertainty Management in Expert Systems," *IEEE Expert*, *5*(2):29–48 (April 1990).
22. Pearl, Judea. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Menlo Park, California: Addison Wesley, 1984.
23. Pearl, Judea. *Probabalistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.
24. Ramoni, Marco and Alberto Riva. "Belief Maintenance in Bayesian Networks." *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 498–505. 1994.
25. Reingold, Edward M., Jurg Nievergelt and Narsingh Deo. *Combinatorial Algorithms: Theory and Practice*. Prentice-Hall, Inc., 1977.
26. Santos, Jr., Eugene. "A Fully Integrated Probabilistic Framework for Expert Systems Development." Research proposal from Air Force Institute of Technology to Air Force Office of Scientific Research Grant #94-0006, 22 October 1993.
27. Santos, Jr., Eugene. "Computing with Bayesian Multi-Networks." Unpublished Report No. AFIT/ENG/TR93-10. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 16 November 1993.
28. Santos, Jr., Eugene. "Efficient Jumpstarting of Hill-Climbing Search for the Most Probable Explanation." *Proceedings of International Congress on Computer Systems and Applied Mathematics Workshop on Constraint Processing*. 183–194. 1993.
29. Santos, Jr., Eugene. "A Fast Hill-Climbing Approach Without an Energy Function for Probabilistic Reasoning." *Proceedings of the 5th IEEE International Conference on Tools with Artificial Intelligence*. 1993.
30. Santos, Jr., Eugene. "Modelling Cyclicity and Generalized Cost-Based Abduction Using Linear Constraint Satisfaction," *Journal of Experimental and Theoretical Artificial Intelligence*, *5*:359–390 (1993).
31. Santos, Jr., Eugene. "A Linear Constraint Satisfaction Approach to Cost-Based Abduction," *Artificial Intelligence*, *65*(1):1–28 (1994).

32. Santos, Jr., Eugene, Solomon Eyal Shimony and Edward Williams. *On a Distributed Anytime Architecture for Probabilistic Reasoning.* Technical Report AFIT/EN/TR95-02, Department of Electrical and Computer Engineering, Air Force Institute of Technology, 1995.
33. Shimony, Solomon E. "Finding MAPs for Belief Networks is NP-hard," *Artificial Intelligence*, *68*:399–410 (1994).
34. Stanat, Donald F. and David F. McAllister. *Discrete Mathematics in Computer Science*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.
35. Sy, Bon K. "Reasoning MPE to Multiply Connected Belief Networks Using Message Passing." *Proceedings of the Tenth National Conference on Artificial Intelligence*. 1992.
36. Sy, Bon K. "A Recurrence Local Computation Approach Towards Ordering Composite Beliefs," *International Journal of Approximate Reasoning*, *8*(1):17–50 (1993).

Vita

Captain Shawn A. Northrop [redacted] He attended Girard High School, Girard, Pennsylvania and graduated as Valedictorian in 1985. He currently holds a dual major Bachelor of Science Degree in Mathematics and Computer Systems from Grove City College, Grove City, Pennsylvania, from which he graduated Magna Cum Laude on 13 May 1989. Upon receipt of his commission, Capt Northrop was assigned to Keesler AFB, Mississippi for technical training as a Communications-Computer Systems Programming and Analysis Officer. On 17 November 1989, he reported to his first duty station at the 2d Space Warning Squadron, Buckley ANG Base, Colorado and served as a Communications Programmer and later as the Chief, Communications Computer Systems. He is a member of Kappa Mu Epsilon National Mathematics Honorary, Omicron Delta Kappa National Leadership Honorary, and was recognized as one of the Top 3 C4 Systems Officers in AF Space Command in 1992. He was promoted to his current rank on 11 June 1993.

[redacted]

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE DERIVING OPTIMAL SOLUTIONS FROM INCOMPLETE KNOWLEDGE BASES	5. FUNDING NUMBERS
6. AUTHOR(S) Shawn A. Northrop	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P Street WPAFB OH 45433-6583	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/95D-08
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr Abraham Waksman AFOSR/NM 110 Duncan Ave Bolling AFB, DC 20332	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Distribution Unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Many real world domains can not be represented using Bayesian Networks due to the need for complete probability tables and acyclic knowledge. However, Bayesian Knowledge Bases (BKBs) are a viable method for representing these incomplete domains, but very little research has been performed on inferencing with them. This paper presents three inference engines for extracting optimal solutions from three distinct BKB subclasses: singly-connected, multiply-connected with mutually exclusive cycles, and cyclic. The singly-connected inference engine has a worst case polynomial run time. Performance improvement techniques for increasing inference engine speed are discussed, in addition to a new tool for measuring incompleteness and aiding in BKB Validation & Verification.

14. SUBJECT TERMS Probabilistic reasoning, inference engines, Bayesian Knowledge Bases, Bayesian Networks, expert systems, incomplete knowledge bases, incompleteness, cyclic knowledge, cyclicity			15. NUMBER OF PAGES 71 / 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA -	Leave blank.
NTIS -	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.